(2)

AL-TP-1991-0008

MEI Associates Inc.
Lexington, MA

AD-A236 800

ARMSTRONG LABORATORY

DESIGNING AN ADVANCED INSTRUCTIONAL DESIGN ADVISOR: POSSIBILITIES FOR AUTOMATION (VOLUME 3 OF 6)

Henry M. Halff

Halff Resources, Incorporated
4918 Thirty-third Road North
Arlington, VA 22207

J. Michael Spector

Department of Computer Science
Jacksonville State University
Jacksonville, AL 36265

HUMAN RESOURCES DIRECTORATE
TECHNICAL TRAINING RESEARCH DIVISION
Brooks Air Force Base, TX 78235-5000

May 1991

Interim Technical Paper for Period August 1989 – February 1991

Approved for public release; distribution is unlimited.

91 6 6 034

91-01421

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5000

NOTICES

[illegible]
J. MICHAEL SPECTOR
Project Scientist

[illegible]
HENDRICK W. RUCK, Technical Director
Technical Training Research Division

[illegible]
RODGER D. BALLENTINE, Colonel, USAF
Chief, Technical Training Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
May 1991

3. REPORT TYPE AND DATES COVERED
Interim Paper – August 1989 – February 1991

4. TITLE AND SUBTITLE
Designing an Advanced Instructional Design Advisor: Possibilities for Automation (Volume 3 of 6)

5. FUNDING NUMBERS

C	–	F33615-88-C-0003
PE	–	62205F
PR	–	1121
TA	–	10
WU	–	43

6. AUTHOR(S)
Henry M. Halff
J. Michael Spector

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Halff Resources, Incorporated
4918 Thirty-third Road North
Arlington, VA 22207

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
Armstrong Laboratory
Human Resources Directorate
Technical Training Research Division
Brooks Air Force Base, TX 78235-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AL-TP-1991-0008

11. SUPPLEMENTARY NOTES
These papers were submitted in conjunction with Task Order #0006 in WU 1121-10-43--they do not constitute the interim or final report.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Advanced Instructional Design Advisor (AIDA) is an R&D project being conducted by the Armstrong Laboratory Human Resources Directorate and is aimed at producing automated instructional design guidance for developers of computer-based instructional materials. The process of producing effective computer-based instructional materials is complex and time-consuming. Few experts exist to insure the effectiveness of the process.

As a consequence, the Air Force is committed to providing its courseware developers with up-to-date guidance appropriate for the creation of computer-based instruction. The assistance should be provided in an integrated automated setting. This paper addresses the nature of the automated setting in which the assistance is to be provided.

14. SUBJECT TERMS
automated design advisor
cognitive learning theory
instructional design theory
instructional systems design

15. NUMBER OF PAGES
88

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

The work reported herein was done for the Advanced Instructional Design Advisor project at the Air Force Armstrong Laboratory (ALHRD -- formerly AFHRL). The substance of this research was done under contract to Mei Associates, Inc., the primary contractor on the Advanced Instructional Design Advisor (Contract No. F33615-88-C-0003).

This work was done as part of the first phase effort on the Advanced Instructional Design Advisor. The initial phase of this project established the conceptual framework and functional specifications for the Advanced Instructional Design Advisor, an automated and intelligent collection of tools to assist subject matter experts who have no special training in instructional technology in the design and development of effective computer-based instructional materials.

Mei Associates' final report for the initial phase is to be published as an Armstrong Laboratory Technical Paper. In addition, Mei Associates received 14 papers from the seven consultants working on this phase of the project. These 14 papers have been grouped into 6 sets and edited by ALHRD/IDC personnel. They are published as Volumes 1 - 6 of <u>Designing an Advanced Instructional Design Advisor</u>:

Volume 1:	Cognitive Science Foundations
Volume 2:	Principles of Instructional Design
Volume 3:	Possibilities for Automation
Volume 4:	Incorporating Visual Materials and Other Research Issues
Volume 5:	Conceptual Frameworks
Volume 6:	Transaction Shell Theory

This is Volume 3 in the series. Dr. J. Michael Spector wrote Sections I and IV. Dr. Henry M. Halff wrote Sections II and III.

SUMMARY

The Advanced Instructional Design Advisor is an R & D project being conducted by the Air Force Human Resources Laboratory in response to an Air Training Command (ATC) Manpower, Personnel, and Training Need calling for improved guidelines for authoring computer-based instruction (CBI) (MPTN 89-14T).

Aggravating the expensive and time-consuming process of CBI development is the lack of Air Force personnel who are well-trained in the areas of instructional technology and educational psychology. More often than not, a subject matter expert with little knowledge of CBI is given the task of designing and developing a computer-based course. Instructional strategies that work in a classroom are often inappropriate in a computer-based setting (e.g., leading questions may work well in a classroom but are difficult to handle in a computer setting). Likewise, the computer offers the capability to present instruction in ways that are not possible in the classroom (e.g., computer simulations models can be used to enhance CBI).

The Advanced Instructional Design Advisor is a project aimed at providing subject matter experts who have no background in computer-based instructional systems with automated and intelligent assistance in the design and development of CBI. The goal is to reduce CBI development time while insuring that the instructional materials are effective.

I. INTRODUCTION (Spector)

The Advanced Instructional Design Advisor is an R & D project aimed at providing automated and intelligent assistance to inexperienced instructional designers who have the task of designing and developing computer-based instruction (CBI). The particular problem being addressed by this line of research is the need for more cost efficient methodologies for the design and development of CBI. Current methods for developing CBI are expensive, time-consuming, and often result in ineffective instruction due to the general lack of expertise in computer-based instructional systems (Spector, 1990).

The Advanced Instructional Design Advisor project is divided into four phases:

Phase 1: Conceptualization & Functional Specifications

Phase 2: Conceptual Refinement & System Specifications

Phase 3: Prototype, Field Test, & Refinement

Phase 4: Technology Demonstration & System Validation

The first two phases have been accomplished with Task Order Contracts. The third phase is being performed via a Broad Agency Announcement (BAA). The fourth phase will be accomplished via a fully specified contract. The work reported herein concerns the first phase.

The next several sections of this paper address the prospects for automating the instructional design process. In considering how to automate this process, a relevant initial set concerns involves the extent to which the process lends itself to automation and various alternative ways of providing that automation. Not every process can be automated. For example, creative processes do not lend themselves to automation since there is seldom an algorithmic representation for these processes.

Processes which involve frequent judgment and adjustment to circumstances may not be good candidates for automation since it is difficult to precisely define how judgments and adjustments are made. Instructional design involves judgment. Difficulty of tasks being trained must be assessed. Appropriateness of instructiuonal strategies to particular lesson objectives must be determined. In addition, adjustments for instructional setting and student characteristics must be made.

Due to these difficulties, it is reasonable to address to what extent the instructional design process can be automated. Halff addresses this question in the second part of this paper. He determines that there are prospects for automation and describes two different approaches: advisory and generative. He recommends the generative approach as the more promising of the two, since it does not address the difficult problem of automating the development of completely new instructional designs.

In the third part of this paper, Halff considers a tentative architecture for the Advanced Instructional Design Advisor (AIDA) and addresses specific problems involved in each of the functional components. He then imagines two incarnations of AIDA. In the first, AIDA is focused on a very narrow subject area and contains more instructional design expertise. In the second, AIDA is designed for broader subject matter domains and is targeted for use by knowledgeable instructional designers.

Halff's two AIDA's are quite different in many essential aspects. The first is built around ready-made instructional templates with default slots. The second is built around a production system and configurable instructional schemata. The first can be used by subject matter experts with little instructional design expertise. The second is intended for use by experienced instructional designers.

The contrast of these two systems highlights the need to decide to what extent human instructional design expertise should be kept in the instructional development process. Halff also emphasizes the need to determine who the users will be and what subject matter areas will be supported by an automated instructional design tool such as AIDA.

II. PROSPECTS FOR AUTOMATING INSTRUCTIONAL DESIGN (Halff)

Introduction

This paper explores both the limits of and opportunities for automation in the design and development of instruction. My intention is to argue for two recommendations in the development of automated instructional design:

1) Automation should assist in the generation of instruction from known designs -- not the creation of designs to be implemented manually.

2) Automation projects should support existing instructional paradigms for broad subject-matter areas (e.g., computer programming), and should not strive to implement some "universal" theory of instructional design.

In the second section of this chapter I lay the groundwork for these arguments by reviewing the main lines of research that do or should influence the field of instructional design. I begin with an examination of theory in learning and cognition, and follow this with a discussion of some newer work on the cultural bases of instruction. Finally, I examine the relationship of these theories to research and recommendations of the instructional design community.

In the third section, I consider two general approaches to the automation of instructional design. On the advisory approach, computers advise instructional developers, and, in particular, devise designs for instructional applications. Design knowledge is represented in a computer and implementation is primarily the responsibility of a human developer. On the generative approach instructional designers rely on computers to generate the instruction that implements an existing design. Design knowledge is primarily resident in a human designer, who uses a computer to help implement the designs. That my sympathies lie with the generative approach should be evident.

In the fourth section of this chapter I describe two challenging but not impossible projects that illustrate this approach. I use these two cases to speculate on the conditions appropriate for automated generation of instruction and on the problem of generality in instructional design.

In the final section, I recap the arguments made in the earlier parts and point the way to more effective automation of instructional design.

Theory and the Prospect of Automation

One of the main themes of this paper is the relationship between the automation of some instructional design function and the justification for that function. Below, I argue that both of these functions require an explicit theory of instructional design. To set the context for the argument, and for other points to be made, a brief review of approaches to instructional design is in order.

Learning Theory

The most natural and most long-standing theoretical approach to instructional design is that of learning theory. From a scientific point of view, the design of systems for human learning should be founded on a knowledge of the fundamental mechanisms of learning. The study of learning mechanisms has always been a major part of psychology. Learning research has its roots in the empiricist philosophers of the eighteenth century; it dominated American psychology almost completely in the first half of this century; and it is manifest today as cognitive psychology.

Fundamental Commitments.

Throughout its history, research and theory on learning have been characterized by two strong commitments:

1) Learning should be understood in mechanistic terms. A complete theory of learning is one that describes stimuli (inputs), responses (outputs), performance mechanisms (rules for generating inputs from outputs), and the laws of learning (rules that specify how the performance mechanism changes with experience).

2) The unit of analysis in any theory of learning is the individual organism. If one can determine the laws that govern an individual's learning, one can then (in theory) account for his or her behavior in any setting, no matter how complex.

An approach to instructional design follows quite naturally from these commitments. The general goal of instruction is to arrange for optimal behavior across a range of stimulus situations. A complete learning theory predicts the behavior that results from any particular set of learning experiences. By inverting this function, it should be possible to determine which experiences optimize behavior.

The Cognitive Revolution.

Early approaches to learning theory were born of the behaviorist tradition. Those working in that tradition proposed that learning was nothing more than the establishment, through reinforcement, of associations between observable stimuli and observable responses. Coupled with performance rules to account for generalization and motivational effects, the behaviorists met with some success both in explaining learning in the lab and in assisting with the design of instruction.

The behaviorists' success, however, was limited to simple S-R spaces and to cases where instructional objectives could be defined in a clear-cut fashion. In the sixties, it became clear, on both theoretical and empirical grounds that the behaviorist approach could not account for the learning of complex skills such as language comprehension. The case of language comprehension is particularly poignant since it can be shown (Gold, 1967) that this skill cannot, in principle, be acquired under behaviorist assumptions. Needed are some additional mechanisms for structuring the learning process.

What emerged from the lessons of the sixties was the realization that most behavior of interest to instructional designers is mediated by intermediate, unobservable cognitive structures, and that these structures are too complex to be induced by a simple behaviorist approach to learning. Associations form during learning not between stimuli and responses but between cognitive structures, and learning typically has as much to do with development of the structures themselves as with the formation of associations among them.

The implications of the cognitive revolution for those interested in a learning-theoretic approach to instruction are not encouraging. Instructional objectives must, at some point in the design process, be represented as the cognitive structures that support skilled performance. Simply denoting the input-output (S-R) requirements of a skill is impossible for some skills and, in the case of other skills, insufficient for instructional purposes.

Unfortunately, there is no consistent methodology for determining the cognitive structures that support skilled performance. A grab-bag of methods such as thinking-aloud protocols can be used to shed light on these structures, and a number of formalisms are available for representing them. However, there is no mechanistic way of applying the methods; there are no generally applicable means of determining the uniqueness (necessity) of a particular cognitive model; and there are no general methods of determining the learning mechanism responsible for the acquisition of cognitive structures.

Although the cognitive revolution did considerable damage to the simple behavioristic approach to learning, it did not change learning theorist's commitment to the fundamental principles mentioned above. Like behaviorists, cognitive psychologists seek a mechanistic explanation of learning, and like behaviorists, they are committed to understanding learning as a phenomenon of the individual learner.

Instruction and Culture

One of the implications of the commitment to the individual learner as the unit of analysis is that uninstructed learning, such as that which occurs when a child walks alone through the woods, is no different in its mechanisms than instructed learning, such as that which occurs when the child walks with a teacher or parent. Instruction can be designed to make the best of the mechanisms of learning, but those mechanisms do not change in nature as the consequence of the instructional situation.

Parsimony argues for the homogeneity of learning mechanisms, but common sense argues against it. Our use of shared knowledge and the instruction that makes shared knowledge possible sets us apart from all other species. It is almost absurd to think that the mechanisms (such as language) developed to share knowledge are specializations of teaching alone and are not accompanied by corresponding specialized learning mechanisms.

In this section we view instruction, not as the manipulation of fixed learning mechanisms but as a cooperative venture, one that takes advantage of corresponding teaching and learning mechanisms. We examine three lines of research, one concerned with cooperative mechanisms for structuring student-teacher interactions, a second that provides information on the role of structure in learning, and a third that examines more global issues concerning the cultural aspects of learning.

The Structure of Classroom Interactions. We begin with Mehan's (1979) "constitutive ethnography" of lessons delivered in a first-grade classroom. His objective was to show that interactions among students and the teacher were governed by a shared "grammar" that defined the structure of a lesson. The model developed by Mehan specified:

the grammatical rules that could be used to "parse" a lesson into its constituent parts and subparts,

the mechanisms used by students and teachers to instantiate the grammar (i.e. to make sure that all parts were included in the right order), and

the mechanisms used to repair the structure whenever it was compromised by external or internal interruptions.

An overview of the grammar itself may give the reader some flavor for the results. Each lesson consists of

an opening that orients the students to the lesson's activities,

a body, in which the class covers the subject matter, and a closing, that serves to inform students of the end of the lesson.

The body, as Figure 1 shows, can, in turn, be decomposed into the treatment or coverage of any number of topically related sets of interactions. Each interaction consists of a query (usually from the teacher), a reply (usually from a student), and an evaluation by the person initiating the interaction. The grammar is flexible enough to allow for some structure in the topically related groups and, more importantly, for extended interactions in which the conversation is extended until all three components satisfy the conversants. Two typical interactions are shown in Figure 2.

Mehan provided empirical support for the model as an account of all of the interactions, captured on videotape, of thirteen lessons on different subjects at different times in a single first-grade classroom. He was able to show that the proposed structure governed the majority of interactions in the classroom and that where the interactional structure was violated, the participants invariably invoked one of the proposed repair mechanisms.

Hierarchical Organization

EVENT	Lesson					
PHASE	Opening		Instructional		Closing	
TYPE OF SEQUENCE	Directive	Inform.	Elicit	Elicit	Informative	Directive
ORG OF SEQUENCE	I-R-E	I-R (E_o)	I-R-E	I-R-E	I-R (E_o)	I-R-E
PARTICI-PANTS	T-S-T	T-S-T	T-S-T	T-S-T	T-S-T	T-S-T

Sequential Organization

KEY: T = teacher; S = student
I-R-E = initiation-reply-evaluation
E_o = evaluation optional in informative sequence

(Adapted from Mehan, 1979)

Figure 1. The Structure of Classroom Lessons

Mehan concludes that success in the classroom depends on mutual adherence to the interactional structure. The teacher found the structuring conventions useful in instructing only because the students "bought into" them. The students found the same conventions useful in learning only because the teacher made use of them in covering the subject matter.

INITIATION	REPLY	EVALUATION
5:77 T: Now who knows what this one says (holds up new card)? This is the long word. Who knows what it says?	A: Cafeteria	T: Cafeteria, Audrey. Good for you.
5:82 T: What does it say over there?	Many: Cafeteria	T: That's right.

(Adapted from Mehan, 1979)

Figure 2. Typical Interaction Sequences.

It is this interdependence that makes it difficult to view what goes on in a classroom from a learning-theoretic point of view. It cannot be said that the students conform to some general laws of learning, for their behavior is clearly fitted to the particular social conventions at work in the classroom. The lessons learned (pun intended) from Mehan's work would tell us little about how people might acquire the same skills in say uninstructed situations or through computer-assisted instruction. Conversely, it cannot be said that the teacher applied general design principles to the construction of her lessons. The devices that she used to cover the subject matter depended on the particular social conventions used in the classroom.

Culture and Learning. After reading Mehan's work, one cannot help but be struck by the author's complete disinterest in learning and instruction. Although he offers elaborate and convincing evidence that classroom interactions are structured by a set of cultural conventions, he makes no claim that the resulting structure has anything to do with what students learn. Mehan's analysis provides much fodder for conjecture on this point, but for a precise account of the role of culture in instruction, we turn to the work of VanLehn (1987) and the design of instruction in multicolumn subtraction skills. VanLehn's early work, done in collaboration with others, is well known and can be easily summarized. The early impetus for research in the area was the observation (Burton, 1972) that children at intermediate stages of learning exhibit systematic error patterns known as mind bugs and that it is possible to give a precise procedural account of each of these bugs. One bug, for example is manifest in a procedure that writes, in the answer row of each column, the result of subtracting the smaller digit in the column from the larger. This bug, illustrated in Figure 3, is called the "smaller-from-larger bug."

$$\begin{array}{rr} & 4\ 1\ 2 \\ - & 3\ 3\ 7 \\ \hline & 1\ 2\ 5 \end{array}$$

(The Smaller-From-Larger Mind Bug)

Figure 3. Subtraction Problem Illustrating Bug

A second step in this research (Brown & VanLehn, 1980) accounted for bugs in terms of a general procedural representation of the multi-column subtraction procedure itself. The procedural representation was rule-based, and the account proposed that bugs arise because students who are "missing" one or more rules are brought to an impasse on certain problems. Confronted with an impasse, the students use a local problem-solving strategy to produce a "syntactically" correct response. The smaller-from-larger bug, for example, is invented by students who have no borrowing rule when they face a problem requiring borrowing.

The third step in this line of research and the one of central concern here, addressed the acquisition of subtraction skills over the four-six year span that they are taught in school. VanLehn reached three crucial conclusions about learning in this situation. First, students acquire the skill by inducing the multicolumn subtraction procedure from exercises and examples. The semantics of the procedure, its relationship to the number system, and all other teleological considerations appear to have no effect on learning. VanLehn based this conclusion largely on the fact that students choice of bugs show no bias towards the semantics or purpose of the procedure.

Second, VanLehn showed that it is impossible, in principle, to induce the procedure for multi-column subtraction from a random sequence of examples or, more precisely, from a sequence of examples taken by the student to be random. The induction problem in this case has two faces. First, the procedure contains disjunctions or choices needed to handle special cases such as borrowing and borrowing from zero. No finite but unstructured set of examples contains enough information to allow a student to discriminate the correct rules. Second, the procedure calls for the computation of intermediate results; the results of borrowing in particular. Examples themselves do not offer enough information to induce the intermediate calculations.

At this point we are faced with a paradox. Students appear to learn this procedure from exercises and examples, yet these exercises and examples are not, in principle, sufficient to support learning. The problem is not specific to multicolumn subtraction. Many of the procedures that we acquire are learned from exercises and examples, contain disjunctions, and have intermediate computations. How then do we learn these procedures?

VanLehn was forced to the conclusion that the exercises and examples provided to the student are not random and were not perceived by students as random. Rather, they are structured in such a way as to permit learning and that this structure was apprehended by the student. He proposed two design principles needed to make the procedure learnable. A one-step-per-lesson rule requires that the exercises and examples in a curriculum be grouped into lessons, and that each lesson addresses a single disjunction or step in the procedure under study. Examples and exercises in a lesson require only the step addressed in the lesson and steps acquired in previous lessons. Second, a show-work principle dictates that the results of intermediate calculations be shown in all worked examples as they are in Figure 4. If these two principles constrain the design of a curriculum and if they likewise constrain the learning process, then it is possible to induce the subtraction procedure from a curriculum of exercises and examples.

$$\begin{array}{rrrr} & 3 & 10 & 1 \\ & \cancel{4} & \cancel{1} & 2 \\ - & 3 & 3 & 7 \\ \hline & & 7 & 5 \end{array}$$

(Illustrating Show-Work, Correctly Worked)

Figure 4. Subtraction Problem With Show-Work

On the design side he showed that textbooks do indeed conform to the one-step-at-a-time and the show-work principles. On the learning side he created all possible learning trajectories admissible under these principles and showed that the bugs and bug patterns in these trajectories matched approximately, but far from perfectly, the bugs and bug patterns found in children's performance.

What is important about step theory is not so much its empirical support as the conclusion that learning mechanisms and instructional design are interdependent. The design principles (one step per lesson and show work) are needed to sufficiently constrain the learning task, but these principles are only effective if they also constrain the student's learning process. Effective instructional design and effective learning are, in this case a cooperative enterprise. The designer and the learner, although without consultation, agree to honor the conventions needed for effective learning. Conventions that thus pervade our society are, by definition, deemed to be part of our culture. In this sense, VanLehn's approach provides the vital insight that culture, at least in some cases, makes learning possible by cutting the induction problem down to size.

Situated Cognition. The implicit presumption in our consideration of both Mehan's and VanLehn's work is that certain cultural mechanisms used for instructional purposes can be abandoned once the student leaves the instructional situation. In other words, the cultural conventions that support the acquisition of a skill can be divorced from the content of the skill itself. More recently, this presumption has been called into question by those interested in the notion of situated cognition.

The roots of the situated-cognition notion can be found in Dreyfus' (1972) arguments concerning the limits of formal (computer) representations as an account for behavior in natural settings. Learning theorists (including cognitive scientists) have long been fond of pointing out that complex behavior is nothing more than the interaction of simple mechanisms with a complex environment. Dreyfus turned this very proposition on its own creators by arguing that the environment is so complex that even complete knowledge of learning mechanisms will be useless in accounting for behavior in arbitrarily-chosen natural settings. The success of any cognitive-science model, Dreyfus argued, rests on simplifying ad hoc assumptions based on the scientist's intuitions about the relevant aspects of the situation being modeled.

More recently, similar arguments have been made by cognitive scientists (Brown, Collins, & DuGuid, 1989; Greeno, 1989) interested in the effect or lack of effect of instruction on cognitive functions in non-instructional settings. Critical to the thinking of these scientists is the contention that non-instructional situations almost always offer contextual support for cognitive operations that completely bypass the methods taught in school to accomplish the same ends. Often cited, for example, is Lave's (1988) observation of an individual who, when faced with the task of obtaining 3/4 of 2/3 of a cup of cottage cheese, measured out 2/3 of a cup, physically divided the result into quarters, and helped himself to three of the four quarters. Because he was situated in a context that afforded him the necessary tools, he was relieved of the burden of formulating a "school" method for achieving the result.

Insights on the situated nature of cognition have two lessons. First, they lead us to the conclusion that the cultural mechanisms used to make learning possible (or easier) may also have the unintended effect of limiting its applicability. In the extreme, the argument goes, instruction is so tailored to the instructional "culture" that it is totally useless in any other culture.

The second lesson is methodological. The methods that we use to study learning and to design instruction are themselves situated. The conclusions that we reach concerning learning and the recommendations that we provide to instructional designers depend on implicit agreements on the nature of instruction. The point is important when we entertain the notion of automating the design process because a computer does not share the same understandings as a human instructional designer. Hence, the same design recommendations may be interpreted in very different manners by machines and humans.

We return to this point in the next section, but the foundation for our discussion there rests on an understanding of the research tradition concerned explicitly with instructional design.

Instructional Design

The instructional design tradition (Gagne & Briggs, 1979) is concerned with the development of a design science for instruction. Judging from the work of this community, their goal is a set of handbooks containing step-by-step instructions for designing, developing, and maintaining instruction from the first inkling of an instructional need to the time when that need ceases to exist.

Those working in this tradition have avoided the commitments of those who proceed from learning theory and have skirted many of the problems of the learning-theoretic approach.

Instructional design researchers are not explicitly committed to mechanism. Many of the guidelines found in their work make no reference to any mechanism of learning (and sometimes have no justification whatsoever). Where learning theory applies, it can be used in design recommendations. Other bases for a recommendation must be used when learning theory is noncommittal or irrelevant to a design decision.

Instructional designers also have been more aware of the situated nature of learning and performance. The research tradition, in fact, receives much of its impetus from the fact that much of training is not oriented towards the job requirements of the students. Instructional designers are exquisitely sensitive to the aims of instruction. They would, recalling Lave (1988) observation recounted above, as a matter of course examine how measuring is done in the job situation and orient instruction to the results of that examination.

Fundamental Commitments. This is not to say that instructional designers do not have their own commitments.

Unlike learning theorists, instructional designers are concerned with the process of designing instruction. They have a fundamental commitment to the development of a uniform design process that covers the life cycle of any instructional enterprise. This commitment has profound implications for both theory and practice. In theory, when considering the balance between the power of skill-specific instructional methods and the power of general instructional principles, instructional designers weigh in on the side of general instructional principles. In practice (and to the annoyance of many consumers of instructional design), instructional designers have a tendency to begin each project anew, often putting forth a major effort only to arrive at a design only slightly different from existing instruction.

A related commitment of instructional designers is to a two-stage process that separates analysis from design and development. That is, they are committed to the belief that instructional objectives can be determined without reference to instructional methods. Conversely (and related to the point made above), instructional methods are not subject-specific. How to teach can be determined by the application of general instructional principles to instructional objectives. What distinguishes the teaching of calculus from the teaching of French are the objectives to be met, and not the principles used to meet those objectives.

This analysis-design process can be found in a number of other fields that espouse a top-down, systems approach to design. The top-down approach in the ISD tradition is typically an exercise in decomposition, classification, and mapping. Primary instructional goals are decomposed into a hierarchy of primary and enabling objectives. These objectives are then classified along several dimensions, and the results of the classification are mapped into instructional methods. Instructional design therefore seeks to advise developers as to the kind of instruction to be used. The content of the instruction is the responsibility of the instructional developer.

The Theoretical Basis of Instructional Design. A theory of instructional design (or any other theory, in fact) need not be correct in all of its details, but it must carry with it the methodology for verifying any assertion made within the theory. We need to ask then what it means to verify any or all of the numerous design recommendations proposed by researchers in instructional design.

The instructional design community has, I believe, serious problems in confronting this issue, and these same problems impose limits on the extent to which the instructional design process, as they view it, can be automated.

The heart of the problem is that the community, while it has been concerned with the development of good instructional design has not been explicit about what constitutes a design to begin with. As it stands now, instructional design recommendations when they work, rely, as do cognitive models, on implicit agreements among researchers and designers on what constitutes a design. The implementation of any particular recommendation in any particular situation depends on the designer's intuitions about how the recommendation should be interpreted in the particular context. In some situations these interpretations are straightforward, however, the variety of contexts and the complex dependencies among instructional objectives make the possibility seem remote that any theory will be able to generate the wide range of human instructional endeavors, and, at the same time preserve all important distinctions among them.

Without knowing the possibilities for any particular instructional design, how those possibilities relate to each other or how the could be implemented, it is impossible to frame a method for validating the recommendation. Even more important for our purposes, the lack of an explicit design space limits the possibilities for automating the design process. Automating the design process itself would require a method for formally representing all of the knowledge that one might want to teach and casting the principles of instructional design so that they could work with the knowledge thus represented.

Approaches to Automating Design

It is not the intent of this paper to explore all of the ways that automation might support instructional design. Instead we concentrate on ways that design knowledge can be incorporated into computers and used to create instructional materials. Eliminated from consideration are the use of computers for support functions such as document processing and the use of computers simply to store and present reference materials on instructional design.

Advisory Approaches

As was mentioned in the introduction, one attractive use of computers in instruction is to involve them in the process of instructional design. A design advisor seems the ideal marriage of the vast body of instructional design rules and current expert-systems technology. Such a combination might well be able to support the design process by

> eliciting, from a human developer the instructional objectives for the application,
>
> eliciting the information needed to classify each objective, and
>
> creating a design for the course by applying design guidelines to the information gathered by the designer.

The advantages of this approach are evident. It would relieve instructional designers of the burdensome bookkeeping associated with the conventional process. It would provide an audit trail that could be used to justify the inclusion or exclusion of material and to provide a basis for course revision. Automating the design process would help to ensure a uniform, presumably high, quality of instruction. Automation would lower the skill requirements for instructional developers.

However, the viability and usefulness of advisory systems rest on certain assumptions, each of which is questionable in the light of the above discussion.

One such assumption is that instructional design principles are separable from content. At issue is the extent to which design principles must be specialized. If the same design principle is used for all objectives requiring, say, classification of individual stimuli, then an advisory system could offer a considerable advantage. If, however, the design of troubleshooting training for, say, Ford Tauri is governed by different principles than that for Buick LeSabres, an advisory system cannot participate usefully in the design process.

The usefulness of an advisory system also rests on the assumption that such a system complements the strengths and weaknesses of human developers. Even if an advisory system could make a substantive contribution to the instructional design process, it may well be that their contribution is not the one really needed by the instructional development community. It is not difficult to envision a system that through an hour's dialog leads an instructional developer to a conclusion that he would have reached unassisted in five minutes. Nor is it difficult to envision the same system leaving the really difficult and time consuming aspects of the development process to the human developer. In short, a system that provides design guidance for human developers is only useful to the extent that those developers lack design skills and possess development skills.

The advisory systems envisioned here are principled. That is, they implement certain principles of instructional design to guide the developer through a top-down design process. The relative usefulness of such an approach depends on the relative effectiveness of this philosophy in dealing with real instructional-design processes. For development de novo, this approach may make sense, but I suspect that little instructional development starts from scratch. New training, in one way or another, is derived from old training. In some cases, developers may draw on the culture of the training institution to create new training. For example, commercial pilots qualifying for a new type of aircraft can expect the same type of training as they received in previous, similar situations. In other cases, the subject matter itself carries its own instructional methods. A scientist faced with the responsibility of keeping her graduate students abreast of their field normally make available to her students the same mechanisms that she used to acquire the knowledge.

Finally, the effectiveness of an advisory system rests on the availability of the information required to drive the course development along the course set by the advisor. Instructional development can and must proceed in many cases when such knowledge is incomplete. For example, a commonly agreed-upon stage in instructional design is the formulation of explicit procedures for successful performance of a task. For some instructional objectives such as multicolumn subtraction, such procedural formulations are available. For others, such as X-Ray interpretation or foreign-language translation, the best that can be hoped for are rules of thumb that only roughly characterize some presumed procedure.

In summary, the usefulness of advisory systems depends on their sufficiency and on the extent to which their functions match the needs of instructional developers. One view of instruction is that of a designed object much like a computer program or spacecraft On this view, advisory systems offer considerable promise since it should be possible to capture effective instructional design in a set of general principles and mechanically apply those principles to create new instruction.

Another view is that instruction is a product of evolution. That is, as knowledge evolves so do the mechanisms for its transmission. Some of these mechanisms are part and parcel of the subject matter; others are cultural; still others are genetic. On this view, the power of advisory systems is definitely limited. The normal course of instructional development is that of adopting and combining existing instructional techniques, perhaps without reference to design principles.

Generative Approaches

A different approach to the role of automation rests on the observation that relatively stable instructional paradigms characterize instruction in a number of areas. Troubleshooting and foreign-language training are two examples taken up below. Data entry, geography, and computer programming are other promising examples. In each of these cases, a general paradigm is configured to fit individual instructional applications. A basic method for troubleshooting training is fitted to different classes of devices. The same can be said for language training and different languages, for computer programming and different programming languages.

These cases offer the opportunity for automation to assist in the generation of materials for broad subject areas. Generative instructional tools already exist in a number applications of computer-assisted instruction. In their simplest form, they generate exercises for drill-and-practice of skills such as typing and arithmetic. In more sophisticated form, they generate instruction from an abstract representation of the knowledge to be taught (Crawford & Hollan, 1983).

Generative approaches tend to be strong where advisory approaches are weak. They can be tailored to the level of generality achieved in the design that they implement. The simple systems for generating exercises in arithmetic are clearly suitable only for arithmetic. Systems like the IMTS (Towne, Munro, Pizzini, Surmon, & Wogulis, 1988) offer instruction across a broad class of troubleshooting domains, and systems such as the CBMS (Crawford & Hollan, 1983) are limited not by the domain but by the form of knowledge in that domain.

In contrast to advisory approaches, generative approaches leave conceptual aspects of instructional design to a human designer and take over the more routine chores of implementing a design. The computer industry has met with considerable success with this basic philosophy for assigning work to people and machines. Generative systems, rather than ignoring the evolutionary nature of instruction, can be tools for accelerating that process. They provide a formal representation of an instructional design, one known to be explicit enough to support the mechanical generation of materials. These designs can be refined, combined, and adopted to different needs. Changes in the resulting instruction can be unambiguously traced back to changes in the design itself. Design principles can be induced and applied within the scope of admissible variations in the design.

Two Proposals for Automated Generation of Instruction

To illustrate the potential for generative approaches to automation and to make some general points about possibilities for automating instruction, I describe in this section two particular opportunities for a generative approach. The applications chosen, foreign-language training and troubleshooting, are intentionally quite different. The first is meant to illustrate what might be done with a well-developed but poorly understood instructional design. The latter is intended to illustrate how general instructional principles, combined with precise knowledge of a skill can be used for controlled, automatic generation of a curriculum.

Foreign-Language Training

A first example of the potential for generative approaches to instruction is oriented towards the achievement of literacy in a foreign language. The situation is of interest both because of what is known about the subject and what is unknown.

Current Practice in Foreign-Language Training. Much of what would be considered essential for instructional design in foreign-language training is simply unavailable. By almost all accounts, literacy in a foreign language is necessary and sufficient for the ability to translate written material between one's native tongue and the foreign language. Unfortunately, we are far from a complete understanding of the mechanisms that might be (much less are) responsible for translation skills. The situation is bad enough that there is no precise standard for an acceptable translation, and, for many texts, genuine disagreement among experts.

Although we lack a precise model of foreign-language literacy or translation skills, we do possess a rough intuitive grasp of the major components of the process. Among these components are

pattern matching to identify idioms and other common expressions,

lexical translation to determine the possible senses and roles of the individual words in the text,

syntactic analysis to determine the the text's phrase structure, and

semantic and pragmatic analysis to resolve lexical and syntactic ambiguities.

Although the details of these processes and the relationships among them are unknown, they serve as the basis for the design of instruction in foreign languages. Typically languages are taught through a sequence of distinct lessons each of which addresses instructional goals relating to idioms, syntax, and vocabulary. (These goals relate to the first three components listed above. Training in the fourth component is not provided on a systematic basis although students are usually required to produce translations that reflect the meaning of the original text.)

Lessons roughly conform to the canons of step theory. A limited number of grammatical constructions are presented so that the grammatical knowledge is built in a stepwise fashion. In addition, texts are chosen that reflect only the vocabulary and grammar under study or that covered in previous lessons.

Opportunities for Automation. In spite of the fact that foreign-language literacy is only partly defined, there is considerable opportunity for automation to assist in the construction of curricula addressing this skill.

At the lowest level, computers could examine candidate texts for each lesson and assemble vocabulary lists. The pattern-matching capabilities of current text-analysis software is also quite capable of identifying idiomatic expressions in text. Hence, given the target text for each lesson, a computer could take over much of the chore of constructing the vocabulary and idiom sections.

More interesting is the possibility that computers could be used to create, refine, or evaluate the syntactic content of lessons. As was mentioned above, part of a typical lesson deals directly with one or a few syntactic components of the target language. Computers are capable of mechanically generating the examples needed to exhibit both well-behaved and irregular examples of the constructions under study.

More importantly, natural-language parsers are well-enough developed that they could both generate and analyze texts to be included in each lesson. Texts used in initial lessons must be highly constrained and are usually generated by the teacher or textbook writer to conform to the limited lexical and syntactic skills of the student. These same constraints should also allow computers to undertake the task of generating sample texts. In later lessons, texts become less constrained and, at some point, are drawn from the existing literature in the target language.

In addition, the task of checking these texts for conformance to instructional objectives becomes more difficult as the students' repertoire grows. Computers could be of considerable value in analyzing candidate texts for their fit into each lesson and even suggesting modifications to eliminate vocabulary or constructions beyond the scope of the lesson. Tools such as these could ensure not only that all objectives of each lesson are covered but also that material going beyond the lesson's objectives is excluded.

Finally, at the most abstract level, computers could assist in the construction of a curriculum that systematically developed the syntax of the language. In particular, it could draw on research in computational linguistics and language learning to develop the kind of procedural representation that VanLehn developed of multicolumn subtraction. Armed with such a representation, it could sequence the components of the procedure, and create exercises and examples that corresponded to the sequence.

Advantages and Disadvantages. I chose foreign language training because it illustrates both the strengths and weaknesses of the generative approach to automation of instructional design.

The possibilities discussed above are not revolutionary. They suggest incremental improvements in both the process and product of an existing instructional design. If that design is fundamentally flawed (as it may well be for training in foreign-language skills), then the mechanisms suggested above will be of little benefit.

By the same token, the approach is wed to the domain of foreign-language training. The computer programs that provide the functions suggested above would be of no use outside of this domain.

On the other hand, in this case where large gaps exist in our knowledge of the target skills, the generative approach to automation provides some promise of improving on existing practices. The development of programs such as those suggested here would ease the development burden, provide improved instructional materials, and sharpen our conceptions of how instruction is generated within the general paradigm.

Moreover, although the approach suggested is not a "universal instruction generator," its generality is well matched to the domain. There may be some set of general instructional principles that, in addition to suggesting the current design or an improved one, will be equally specific and helpful in its suggestions for history and automobile-repair curricula, but such design tasks are far from automatic given the current state of research on learning and instruction.

Troubleshooting

Our interest in foreign-language training was based on the existence of an instructional design applicable to a bounded but usefully large set of specific instructional needs. A general paradigm for troubleshooting training also exists, but the the paradigm itself is far less structured than that for foreign-language training. Typically troubleshooting training is device specific. Students are, through lectures, provided with the theory of operation of a device and perhaps some hands-on practice manipulating the device. The heart of troubleshooting training is a sequence of exercises in which students must isolate a single specific fault in the device or a simulation thereof. The faults used in these exercises are typically chosen on the basis of their importance, and sequenced in order of difficulty. Importance and difficulty are typically decided on the basis of subject-matter experts' intuitions.

Our psychological knowledge of troubleshooting is far more developed than that of language learning. The performance requirements of the skill are well known, and we have a reasonably complete picture of the cognitive concomitants of the skill. It should therefore be possible to construct models of particular troubleshooting tasks and use these models to select and sequence exercises.

Troubleshooting Defined. Troubleshooting, for our purposes, is the identification of faulty components in malfunctioning equipment. Our view of equipment is deliberately simplified. We view a piece of equipment as a network of components. Each component at any time is in one or a number of possible states. Each of the components receives inputs from one or more other components and delivers outputs to other components. Each of these outputs is a function of the component's inputs and state. Troubleshooters can observe some of the outputs and the states of some components. They can manipulate the states of some components (e.g., switches). Costs can be assessed for observations, replacements, and panel manipulations.

In typical training situations, certain simplifying assumptions govern the behavior of the equipment.

> Every malfunction is the result of a single faulted component, although in real equipment multiple faults often occur.
>
> Faults can be characterized as a change in the state or possible states of a component, not in the topology of the equipment, although in real equipment faults can change the nature of the connections among components.
>
> Neither testing nor replacing a component will fault another

component, although in real equipment a faulted component can protect another component from damage.

Finally, we assume that there are no faulty replacements, even though real world technicians will on occasion return a faulted component to inventory.

These restrictions are the ones traditionally used in troubleshooting training and in tests of troubleshooting competence. I suspect that they are part of the maintenance-training culture. Without them, many troubleshooting exercises would be insoluble and many of the soluble ones would be uninstructive.

In some settings, other simplifications may also apply. For example, feedback loops may be eliminated, or components may be limited to a single fault mode.

The Cognitive Psychology of Troubleshooting. Cognitively oriented studies of troubleshooting are not new, however, it is only recently that we have seen theories of skilled troubleshooting sufficiently precise to support modeling of individual performance.

One compelling account of troubleshooting skill is that of Rouse and his colleagues (Hunt & Rouse, 1984; Rouse & Hunt, 1984; Rouse, Rouse, & Pellegrino, 1980). A recent version (Hunt & Rouse, 1984) of this group's theory holds that skilled troubleshooters work with two distinct strategies. Both of these strategies are represented as sets of rules that control the focus of attention in troubleshooting, observation and replacement decisions, and deductions made on the basis of each observation's outcome. One set of rules, called T-rules, captures device-independent troubleshooting expertise. These rules match configurations found in the device to common patterns which are, in turn, associated with troubleshooting actions or decisions. Hunt and Rouse (1984) give the following example.

IF the output of X is bad and X depends on Y and Z and,
IF Y is known to be working, THEN check Z.

The rule potentially applies to any three components of the device that match the rule's condition.

A second strategy is embodied in rules called S-rules that capture device-specific trouble-shooting skills. S-rules match to specific patterns of observations in the device rather than general configurations of components. Rouse and Hunt suggest the following S-rule for troubleshooting automobiles.

IF the engine will not start and the starter motor is turning and IF the battery is strong, THEN heck the gas gauge.

Both S-Rules and T-Rules are local in scope. That is, their conditions contain no information on the overall impact of applying the rule. For example, the T-Rule given above might match a part of the device that contains fairly little information (in the information-theoretic sense) about the fault's location and also to a pattern containing a great deal of information about the location of the fault. All other things being equal, troubleshooters should, and do, choose the latter configuration. Rouse and Hunt found it necessary to account for this effect by conditioning the rule selection process on each rule's usefulness. Towne, Johnson, and Corwin (1983) also found that observations are chosen largely on the basis of their information value.

The work discussed above can be summarized in the conclusion that skilled troubleshooters base their troubleshooting decisions on three aspects of the troubleshooting situation:

device-specific associations between symptoms and faults

device-independent configurations of components, and

the information-theoretic value of potential decisions.

One final result worth mentioning concerns the generality of trouble-shooting skills. Higher-level cognitive skills have been found to be generally resistant to transfer, but the results described in Rouse and Hunt (1984) indicate that if students are taught to troubleshoot a variety of different systems, they will have an advantage in learning to troubleshoot a new system. Two factors, I believe, were critical to this finding. First, Rouse and his colleagues knew where to look for transfer in the sense that device-independent techniques (T-rules) are known to operate in the training and transfer domains. Second, the successful transfer experiments provided training in more than one domain, thus conferring an advantage on device-independent skills during training.

<u>Computer-Aided Generation of Troubleshooting Curricula</u>. The foregoing suggests that troubleshooting training can have either or both of two distinct objectives. Some individuals (consumer electronics technicians, for example) may need training in device independent troubleshooting techniques. Others (advanced avionics technicians, for example) will need training in device-specific skills. The training techniques for these two objectives will be different, but computers can help in each case.

In this section, we suggest principles for selecting and sequencing troubleshooting problems (as exercises and examples) and the role of computers in implementing these principles.

1. Teaching General Troubleshooting Skills

The results cited above indicate that general troubleshooting skills should be taught by using problems drawn from a variety of devices with a variety of structures. The problems should be selected to promote cognitive skills corresponding to the three strategies listed above:

pattern-recognition skills that enable troubleshooters to identify simple configurations of components,

skills in identifying the appropriate troubleshooting action for each of the patterns identified, and

device-comprehension skills that allow students to choose the most information-laden observations.

With certain extensions, current models of troubleshooting, such as that of Hunt and Rouse, could be used to provide specific definitions for each of these skills. The main stumbling block to achieving such a model at this time is the lack of a suitable representation scheme for devices. Both intuition and evidence from studies of cognitive structures indicates that troubleshooters represent devices in terms of a hierarchical decomposition. Such a decomposition makes device comprehension manageable and almost automatically leads the troubleshooter to the most information-laden observations. Also, to the extent that the device is designed hierarchically, a decomposition may be evident in its documentation.

The task of creating a curriculum of problems is one of finding the principles that apply to this situation. Note, for example, that the learning situation envisioned here is subject to the same sources of difficulty that VanLehn found in multicolumn subtraction. Both the pattern-recognition and device-comprehension skills involve disjunctions (choice points), and unobservable intermediate results (the choice of a pattern) are involved in every troubleshooting decision. This suggests that problems be grouped into lessons that reflect the stepwise development of pattern-recognition and device comprehension skills and that some method be used to exhibit the intermediate results required to make each troubleshooting decision.

The availability of a psychologically valid model of device-independent troubleshooting skills makes it possible to represent the design requirements for the curriculum in formal, computational terms. Thus, a computer could be used to candidate problems for each lesson that meet the stepwise refinement requirements of the lesson. Furthermore, given a suitable way of displaying the results of intermediate results, computers could generate materials that would show students otherwise unobservable steps in the troubleshooting process.

The system sketched above could not, in itself, completely automate the generation of instructional materials. For one thing, the system would not generate a unique curriculum, and developers would need to choose among those available. In addition, giving a nod to proponents of situated cognition, some effort should be made to situate the abstract problems in realistic scenarios. Finally, it might be wise to include a verbal description of the troubleshooting procedure and its basis.

2. Teaching Device-Specific Troubleshooting Skills

Device specific troubleshooting skills are those that rely on recognition of specific patterns of device behaviors, such as the pattern exhibited in the S-rule provided above. The research discussed above indicates that these recognition skills tend to develop naturally as students become familiar with the devices. Of interest here are the possibilities that training can accelerate the acquisition of device-specific strategies and, more importantly, selectively promote the acquisition of more effective strategies.

The key to realizing these training goals is to develop a specific set of device-dependent strategies; determine the troubleshooting procedure (typically a discrimination net) that best implements the strategies; and devise a curriculum targeted to that procedure. Computers can help in a number of ways.

Computers can generate alternative device-specific strategies.

Computers can evaluate the relative utility of a device-specific strategy by comparing its cost to that of its device-independent counterpart.

Computers can evaluate an entire set of device-dependent strategies by inducing the troubleshooting procedure needed to implement the set and assessing the complexity of the procedure.

Computers can generate the sequence of problems that will most effectively teach a device-dependent troubleshooting procedure.

3. Keys to Generative Curricula

We can summarize the suggestions made above by recapping the philosophy behind automated generation of troubleshooting curricula. Automation becomes relevant in this case because of the potential for formally representing the space of troubleshooting problems in a way that reflects the knowledge requirements of the skill. It is under these circumstances that instructional design principles, such as show-work, shed their vague character, and become tools for automatic curriculum generation. In both of the cases discussed above, we envision a system that, working from a procedural representation of the target skills, creates a sequence of troubleshooting lessons. The first lesson would start with the simplest case. Subsequent lessons would add refinements to the procedure, one step at a time, by introducing problems that exercise the use of the target refinements. In addition, for each problem generated, the computer would provide a trace of each problem's solution. These traces could form the basis for implementing the show-work principle.

The important point of this example, however, is not the extent to which instructional development is automated but the nature of the automation and its prerequisites. Required are a formal and psychologically valid model of the troubleshooting process and instructional principles that can be applied to the model. The result is a curriculum guaranteed to conform to the principles. This guarantee cannot be met if one of the three components -- formality, psychological validity, or instructional principles -- is missing.

Is There a Single Generative Approach? The two examples presented here are, by design, quite different. The foreign-language example shows that automation has the potential for generating and evaluating materials even when the target skill is not well specified. The opportunity for automation in this case is the existence of a well-worked out instructional design whose components can be matched with formal cognitive representations of the material needed to implement the design.

The second example, troubleshooting training, by contrast suffers from a weak instructional design and fairly complete knowledge of the skill itself. In this case instructional principles can be put in correspondence with a formal cognitive model of the target skill in order to produce a curriculum.

Thus, to the extent that we accept the promise in these approaches, it appears that there is no one set of conditions that make a generative approach attractive. The strength of the generative approach is its ability to achieve uniformity within its area of application. Its application to any one area may be almost completely ad hoc.

I suspect that the ad hoc nature of this approach has to do with the nature of instruction in general. In both of the examples above, some formal representation of cognition in the target area corresponded to some set of instructional principles or methods. Since I see no prospect for a uniform, formal interface of all instructional methods to all domains of human knowledge, opportunities for generating instruction with computers will remain a matter of judgement and intuition.

Conclusions

The discussion above has pointed out some of the opportunities for and limitations on automation in instructional design.

In our discussion of theory, we saw that conceptions of instruction have developed from a mechanistic learner-centered approach to one that recognizes the importance of cultural mechanisms in both the acquisition and exercise of skill. The picture that emerges of the instructional enterprise is one more of evolution than design. The methods for transmitting knowledge are part and parcel of the knowledge itself. As a bit of knowledge changes, interacts with other bits, and assumes different forms in different contexts, the instructional methods that keep the idea alive will change in corresponding ways. (See Dawkins (1976) for a discussion of related ideas.)

The implications of this view for the instructional design community are relevant to the question of automation. The effectiveness of top-down instructional design methods is not at issue; they clearly work when applied appropriately in the appropriate situations. What is of interest is the reason for their effectiveness. I contend that the key to their success is the human designer, who can interpret general recommendations in specific contexts. Because humans can appreciate the dependencies among different aspects of a domain of instruction, and because they can work out the implications of these dependencies for an instructional design, the presence of people in the loop guarantees that the complexity and variety of instructional designs will be sufficient to meet the complexity and variety of domains to be taught.

Computers, however, have none of the context needed for effective design of instruction. The uniform, general implementation of a body of instructional design principles in a computer, would require a uniform, general method of representing all that might ever need to be taught. Also needed would be an interface between this representation and the body of instructional design principles. Neither of these components is within the grasp of current approaches to knowledge representation.

These conclusions lead us to some particular recommendations regarding automation in instructional design.

Computers Should Make Things Better or Easier

The appropriate use of computers is for tasks that humans do poorly or with some difficulty. We have suggested above that one such class of tasks is the generation of large amounts of instructional materials that must, by design, conform to complex specifications. To the extent that those specifications can be formalized, a computer can be used to automate the process.

Other opportunities for automation exist in roles supportive of the instructional design process. These include document processors, authoring systems, data-analysis programs, and automated retrieval of reference materials, to name a few. One could even envision an automated procedural guide to the instructional design process that would track the design as it developed and make suggestions for completing the design.

Not useful would be any tool that pre-empted design tasks that could be done better and more easily than an unassisted human developer. A program that insisted on wending its way through a complex set of design rules and principles on the pretext of actually creating a design would, for reasons cited above, be a counterproductive use of computers.

Build on Existing Instructional Knowledge

If, as I propose above, instruction develops more by evolution than by design, it would be a serious mistake to provide computer support for one design approach without examining the potential for supporting other approaches. Many courses developed today are, either explicitly or implicitly, derived from other similar courses. This imitative approach to design may be inappropriate in some instances, but in many cases (and not just in instruction) it is the most effective route to a new design. More than passing attention should be given to automation efforts(like the projects suggested above) that support the adaptation of a design to many different applications. Not recommended would be a commitment to support with automation only those design efforts that conform to a top-down ISD process.

Generality is a Sacred Cow

The instructional design community has made no bones about its commitment to offering guidance on the development of every sort of instruction. This commitment to generality is laudable on general grounds and makes particular sense in the context of instructional design. One reason for embracing generality is that one never knows what sorts of instructional mechanisms will be needed until the design process is well underway. The general stance is also a welcome contrast to typical efforts in other research communities. More often than not educational research deals with effectiveness of a single mechanism in a single, often artificial context. Real instructional problems are only solved with a combination of instructional techniques. One of ISD's major strengths is its ability to help designers assemble and configure a variety of mechanisms to meet particular instructional needs.

However, the goal of generality in the design process itself is not necessarily a recommendation for generality in automated support for instructional design. Todays computers are simply not equipped to make many of the judgements needed to create an effective design. They can offer support for selected tasks in the design process and they can participate, in certain cases, in the generation of materials. Those interested in supporting instructional design with computers should pitch the level of generality of their efforts to those appropriate for the tool being developed, not at the level of generality of the design effort itself.

It is my hope in this chapter to have introduced some moderation and humility into efforts at automating instructional design. It may help to recall that we, almost without knowing it, are the consummate teachers and students of the animal kingdom. We have, depending on how you count, several million years of collective experience in teaching and learning from each other. Nearly every communicative act among us has some instructional aspect. Hence, it should come as no surprise that we face some difficulty in expressing our knowledge of instruction in the precise, formal terms needed for computer implementation.

In the next chapter, however, I shall attempt to sketch two specific ways that instructional design expertise may come to reside in computer-based systems. The two systems I shall outline are as divergent as the two approaches to automating instructional design described in this chapter, although both represent generative approaches to the problem, and, as a consequence, are genuine possibilities to pursue.

III. AUTOMATING INSTRUCTIONAL DESIGN AND DEVELOPMENT (Halff)

Introduction

This paper is about instructional design and development, its implementation in different contexts, and opportunities for its automation. I begin by describing a general conception of instructional design, development, and delivery. This conception (called the AIDA concept) is nothing more than a simplified description of current thinking about instructional design. The AIDA concept was developed to guide development of automated tools for instructional development and design.

(AIDA is the name of a Egyptian slave and gifted soprano, who was buried alive with her lover Rhamades. In more recent times, the term is used to refer to an Automated Instructional Design Assistant. It is too soon to tell whether the modern-day AIDA will meet the fate of its ancient namesake.)

My main intention is to examine how the AIDA concept applies in particular instructional development contexts and to suggest how the Concept could be most profitably automated in those contexts. This paper pursues that intention by elaborating the AIDA concept in two specific instructional development contexts (one real, one fictional). The discussion of each context includes a thumbnail sketch of its main characteristics, a description in terms of the AIDA concept, and suggestions for appropriate automation of the Concept.

As will be seen, certain important differences between the two contexts have a critical influence on the automation of instructional design within each context. By way of conclusion, I summarize these features and discuss their impact on further development of AIDA.

The AIDA Concept

The AIDA concept, illustrated in Figure 5, is a description, in information processing terms, of instructional design, development, and delivery currrently being proposed as the high level AIDA architecture. A summary of the figure may help orient the reader to the following, more detailed account.

In this system, there are two primary inputs to the instructional design effort:

SET -- information about Students, the learning Environment, and the Task(s) to be mastered, and

Content -- a specification of the material to be mastered by the students.

These two inputs are submitted to the AIDA Executive, a central component of the AIDA concept. The Executive is a function that maps the space of {students} X {environments} X {task} X {content} into a set of instructional strategies.

These strategies are not complete instructional regimes, but rather procedures that are executed or interpreted to provide instruction.

<u>Instructional delivery</u>, the ultimate output of the AIDA concept, is the process of running the instructional strategy to deliver instruction to the student.

The AIDA concept (designed, as it was, by a committee) is a weak one, embodying only the most fundamental principles of instruction:

> that instruction should be sensitive to students, the learning environment, the task to be learned, and the course content,
>
> that some level of abstraction is possible in the design of instruction, and
>
> that effective instruction must eventually find its way to the student.

Therefore, further specification is required before any serious use can be made of the concept. The bulk of this section discusses the major issues associated with the components of the AIDA concept and the interrelations among them.

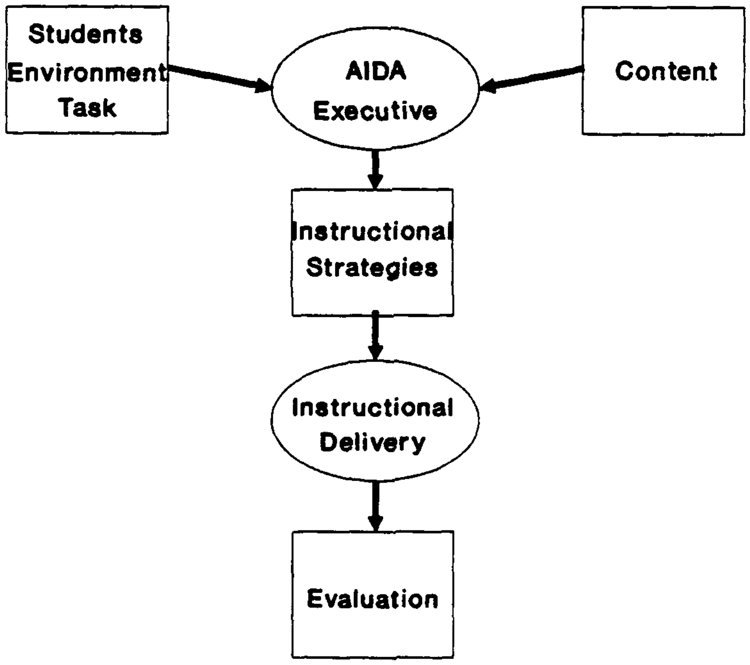

Figure 5. The AIDA Concept

The Content Component

In describing the Content component, we need to ask about the function of the component, about its structure, and about the practices used in particular instructional situations. All three of these issues are problematic.

A naive behavioristic view holds that content is nothing more than a description of the tasks that students should be able to perform as the result of training. Views informed by cognitive science, however, see the futility of this approach for all but the simplest of instructional situations. Most instructional developers now view course content more in terms of what students are to learn.

The switch from a behavioral to a cognitive perspective provides two advantages. First, it allows one to precisely represent complex instructional objectives, for example, that of recognizing a well-formed chemical formula. Second, it incorporates into the instructional design process, the specification of the intermediate cognitive structures that support skilled performance. For example, a behavioristic specification of the content of a troubleshooting course might be nothing more than a list of symptoms and appropriate troubleshooting actions (tests and replacements). A cognitive representation would include specifications related to interpreting block diagrams, selecting tests, and refining hypothesis lists.

Unfortunately, the more ambitious one is about the function of the Content component, the more difficult and complex are the problems of specifying its form and implementing that specification. Most instructional designers seek a language-like representation of instructional content in which structure (syntax) is clearly separated from content (semantics). This separation finds its advantage in principles of instructional design that operate on the structure or syntax of the content without having to worry about the content or semantics. In practice, the representation of content is even further constrained by instructional-design considerations. Most views of instructional design (e.g., Gagne and Briggs, 1979) require a content representation in terms of instructional objectives that can be matched to corresponding instructional methods.

Once the function and form of the Content component have been adopted, there remains the issue of actually representing a particular content area. Neither the cognitive science community nor the instructional design community has had much success in devising powerful, uniform, broadly applicable methods for inducing the cognitive structure of a skill. What we have instead is a grab-bag of empirical and formal methods with only the roughest of intuitive guidelines as to their use. A reasonably sharp cognitive or instructional scientist can usually, by dint of intensive effort, create a cognitive representation of a skill. However, another individual would probably arrive at a completely different analysis, and no one would find either of their analysis to be of much help in conducting an analysis of a different type of skill. The most hopeful statement that one can make is that skills may come in families, for example, electronic troubleshooting, foreign-language learning, and computer programming. Analysis of the family as a whole may yield significant dividends over separate analysis of individual members.

The following critical points summarize the foregoing discussion:

Content must be represented in a generative, cognitive form. It is a competence model and must, of necessity, identify the mechanisms whereby competence is to be achieved.

Instructional design methods place non-trivial constraints on the form of the Content component. They require that syntax or structure be separable from content or semantics. They also require the explicit representation of instructional objectives in the Content component.

Even within these constraints, there are no uniform methods for analyzing particular skills. Some skills may, however, fall in families that admit to a common structure.

The SET Component

The SET component represents information about Students, the learning Environment, and the target Tasks. Different conceptions of the structure of this component lead to different approaches of the instructional design process.

The simplest conception of the SET component is that of a fixed, finite-length list of features that describe certain aspects of the students, instructional environment, and target tasks. The list might include level of motivation (a student feature), availability of laboratories (an environment feature), and availability of job aids (a task feature). The feature list could be quite long, but only a fraction of its members would be relevant to any particular design.

The feature-list approach to SET leaves much to be desired. In particular, it does not allow for use of content-specific aspects of students, the instructional environment or the task. Not represented in a fixed feature list, for example, are the student's level of mastery of particular objectives, the availability of equipment for practicing particular parts of a skill, or the frequency with which particular operations are encountered on the job. Accommodating this information would, at the least require an overlay approach (Carr and Goldstein, 1977), whereby SET features are represented in a structure isomorphic to that of the Content component. Thus, if the AIDA Concept is to produce instruction sensitive to particular student knowledge and competence, particular training opportunities, and particular task characteristics, the SET component must be dependent on the Content component.

Also problematic is that aspect of the SET component that is innocuously labeled "task." As originally conceived, this aspect of the SET component specifies the goals and behavioral objectives of the skill to be learned. It therefore forms the basis for the content analysis described above. In most cases, the task may itself have a complex structure, one that is imposed on the Content component.

The relationship between students, tasks, and content can be even more complex. Consider, for example the following three situations:

1. Students proficient in the maintenance of avionics systems in general are to be trained to maintain the avionics of the F-22, the Air-Force's latest fighter.

2. Students not proficient in the maintenance of avionics systems in general are to be trained to maintain the avionics of the F-22.

3. Students not proficient in the maintenance of avionics system are to be trained in to maintain the avionics of the F-22 in such a way that they also acquire skills helpful in or sufficient for maintenance of other avionics systems.

These three cases, in which the primary instructional objectives are ostensibly identical, each call for vastly different representations of student knowledge, task definition, and content knowledge.

To summarize, the separation of SET from Content shown in Figure 1 may be misleading. Whenever content-specific aspects of the students, environment, or task play a role in the instructional design, the SET component will acquire a structure derived from the Content component. Conversely, material in the Content component may, in part, derive from the structure and/or content of the task description. Thus, in some instructional designs, the SET and Content components will be functionally independent; in others they will be so closely related as to be inseparable; and one can envision the full range of possibilities between these two extremes.

Instructional Strategies

An instructional strategy is a procedure for teaching a single objective. It is abstract in that it is general over the class of objectives to which the strategy applies. It takes, as data, student characteristics and the material that defines the particular objective. It produces, as output, instruction to be delivered to the student.

A couple of examples may help make the concept clear. Figure 6 (below) presents a strategy for teaching a serial, non-branching procedure. Note that it is applicable to any serial, non-branching procedure. (However, I make no claims for its effectiveness or even the adequacy of the representation.) Figure 7 (below) illustrates a strategy for teaching computer programmers how to use a primitive of a new language when a corresponding primitive of an old language is available. Both strategies are general. However, that of Figure 6 is applicable to a wide range of domains; that of Figure 7 is applicable to one domain and only in particular circumstances. The strategies employed in most curricula are a mix of general and domain-specific strategies.

Also worth noting here is that the strategies in Figures 6 and 7 make no commitment to media or other aspects of the training environment. Depending on the interpretation of terms such as elicit and exhibit, one could execute these procedures in a classroom, in a laboratory, or on the job.

The representation of instructional strategies is a critical issue for ISD in general and for the AIDA Concept in particular. To appreciate the enormity of the problem, it is helpful to remind ourselves that a complete representational theory of instructional strategies must accommodate both the vast range of material that can be conveyed through instruction and the vast range of mechanisms available for conveying material.

As Merrill (1989) points out, the instructional design research community is of two minds on the nature of strategies. The conventional approach, which he calls "ID-1," is based on a set of universal instructional primitives that can be composed into strategies. One primitive, for example, might produce a verbal statement of some aspect of the material. Another might generate and exhibit an example. Others might query students in particular ways.

```
OBJECTIVE
    Error-free execution of a serial,
      non-branching procedure
STRATEGY
    Inform the student of the nature of the objective
    Exhibit the entire proceudre
    For each step, N,
        Repeat until success
            Exhibit steps 1 to N
            Have the student execute steps 1 to N
            Success is error-free execution
```

Figure 6. Instructional Strategy -- Non-Branching Procedure

```
OBJECTIVE
    Use of a programming primitive, p,
      in a new language, L
PREREQUISITE
    Use of the corresponding primitive, p',
      in an old language, L'
STRATEGY
    Exhibit the syntax and function of p
    Exhibit the relationship of p to p'
    Exhibit a procedure in L' using p' and
      the corresponding procedure in L
    Exhibit a program in L' using p' and
      elicit the corresponding program in L
    Exhibit a programming assignment in L that
      requires the use of p and elicit the solution
```

Figure 7. Instructional Strategy - Language Primitive

One might argue that the strategy illustrated in Figure 7 is nothing more than a specialization of some domain-independent strategy. A proponent of this argument, however, would be obligated to exhibit the operations that permit some strategies to be represented as specializations of others. She would also be obligated to show, in this case, which generalization of the strategy in Figure 7 is most useful in designing computer-programming instruction.

Merrill (1989) and I (Halff, 1989) have pointed out that an approach based on primitives leaves much to be desired. In the first place, the set of primitives is open ended; there is no principled way of generating a unique, or even a canonical set. Second, effective instruction incorporates complex constraints among instructional primitives as they are assembled into strategies. A teacher, for example, may refer to particular exercises in a textbook. Exercises and examples in most curricula are subject to sequential constraints (VanLehn, 1987). Interactive instruction also appears to be governed by complex relations among individual instructional primitives (Collins & Stevens, 1982).

Needed, therefore, to compose primitives into strategies, are not only the mechanisms for selecting the primitives but also ways of representing procedurally all pertinent constraints among them. Needed, therefore, is nothing less than a generative grammar of instruction. Although it may be possible to write such a grammar for particular objectives or fixed set of objectives, to actually write one for an interestingly large class of domains is well beyond any foreseeable advances in instructional design. This is not to say that skilled instructors do not or cannot implicitly exercise such a grammar in the course of designing instruction.

Rather than dealing with instructional primitives, it therefore seems more promising to collect and implement instructional strategies as integral, black-box units, which Merrill (1989) calls transaction frames, a term that I will continue to use. On this approach, strategies can be tailored to individual situations in specific, ad hoc ways, but they cannot be disassembled, transformed, and reassembled. Once configured to a particular instructional purpose, transaction-frames become special-purpose procedures called simply transactions. Transaction frames or black-box strategies as black boxes can be viewed as cultural products, subject to the methods and principles of natural science. Even if they cannot be generated, they can be collected, classified, studied, and used.

What one gains from the transaction-frame approach is feasibility. What one sacrifices is generality. By simply collecting all of the transaction frames appropriate to an objective or domain, one can ensure effective instruction for the objective or domain. There is no principled way of applying the products of this exercise to another domain. Different objectives will almost certainly require different transaction frames, and some, all, or none of the instruction frames appropriate to one domain could be useful in another.

Neglected in the current conception of AIDA is the notion of a curriculum, the collection of strategies that generate a course of instruction. I mention it only in passing here, noting that the problems that arose at the strategy level arise also at the curriculum level. Curricula must reflect complex, often ad hoc, constraints among strategies. The two general approaches to representing curricula are (1) a generative grammar in which strategies are lexical items or (2) ad hoc existing schemata that dictate the configuration of strategies in a curriculum. (See Reigeluth and Stein (1983) and Gagne and Briggs (1979) for suggestions regarding the first approach.)

In summary, we have been concerned with the instructional strategies that constitute procedures for delivering objective-specific instruction in the AIDA concept. Strategies might be represented by a generative scheme based on a fixed set of instructional primitives and a grammar expressing the structure of strategies. A more feasible alternative views strategies as black boxes (transaction frames) that can be tailored on an ad hoc basis to meet particular instructional objectives. The generative approach has the advantage of providing a complete, principled approach to strategy representation but is infeasible for general instructional design purposes. The transaction-frame approach is potentially labor intensive and not based on explicit principles but it does offer extensibility to almost any domain.

The AIDA Executive

The task of the AIDA Executive is, in brief, to analyze any of a broad range of Content and SET and to produce a set of procedures that will effectively teach the material. The Content can can be represented in any number of ways: semantic nets, production systems, uninterpreted text, and neural networks, to name a few. The procedures created by the AIDA executive must be able to transform these various representations into forms suitable for consumption by students, human instructors, and mechanical teaching devices.

The procedures must also, when run, provide effective instruction. The AIDA executive therefore has a problem somewhat more challenging than that of software design, and the central issue in the design of the Executive is how to meet this challenge.

What makes the task feasible are constraints on the representation of both the input to and output from the AIDA

Executive and a judicious division of labor among the subcomponents of the Executive:

> The content is represented by a grammar that parses the subject matter into objectives.
>
> Instructional strategies are either generated or selected and matched to objectives.
>
> The strategy is configured to the content by filling slots for content-specific material.
>
> The strategy is configured to the student(s) and the training environment by filling slots with corresponding information from the SET component.

Thus, the the Executive's success lies in properly structuring the Content and SET components and providing a workable set of strategies. If strategies are generated from instructional primitives, then the Executive composes each strategy by exercising the generative grammar in the context of content and SET specifications. If strategies are based on transaction frames, then the Executive selects the frame and configures it according to content and SET specifications.

Instructional Delivery

The reader at this point may ask why the AIDA Concept makes reference to strategies at all. Why not connect the AIDA Executive directly with the student by making it responsible for instructional delivery? The answer to this question lies in the need for interactive instruction. Because instruction is generally interactive, the product of an instructional design must have a procedural representation. That is, it must be represented as a strategy.

This said, it behooves us to consider the nature of interactivity in instruction. Current instructional design methods can be classified into one of a small number of levels of interactivity:

> Some forms of instruction are non-interactive. Uninterruptable lectures, films, and other presentations are of this sort.
>
> Most forms of instruction offer at least a moderate level of self pacing. Students can decide when to turn the pages of a text or can proceed through the steps of a laboratory exercise at their own pace.

Most forms of instruction offer some sort of local feedback. The practice of many skills (e.g., bowling) automatically provides information about performance.

Some forms of instruction offer local interactive control over the curriculum. The branching strategies used in most computer-based instruction are of this sort. I use the term local here to indicate that branching decisions in this type of instruction are context free.

Qualitative simulations such as STEAMER (Hollan, Hutchins, and Weitzman, 1984) are examples of instruction that reach a context-sensitive level of interactivity. The response of such systems to student actions is a complex function of the evolving instructional context.

Some forms of instruction are conversational. They rely on strategies that query, inform, and listen to students in natural language or a medium of equivalent power and complexity. Classroom and tutorial discussions are prime examples of this level of interactivity. These forms of instruction are not only context sensitive, but also incorporate some form of planning and abstraction of the instructional context.

These levels roughly delineate the information-processing requirements of the instructional delivery mechanism(s). Self-paced instruction, for example, need only be able to determine when to undertake the next step in an instructional procedure, based on input from the student. The mechanism for delivering conversational instruction, on the other hand, must have the full power of natural language, that is, something beyond the expressive power of a context-free grammar.

Also important is the interface between the instructional delivery mechanism and the curriculum. Instructional delivery systems that embody complex instructional procedures can operate with high-level representations of the curriculum. Competent human instructors, for example, can convert general written guidelines for a classroom discussion into a complex procedure for conducting the discussion. Strategies for computer delivery need to be written in machine-interpretable form. Some forms of instruction such as text and (uncoached) batting practice need no interpretation at all.

In summary, the design of an instructional delivery system must take into account two types of information-processing requirements. First, the level of interactivity with students will impose information-processing demands on the delivery system. Second, the delivery system must be able to interpret strategies in the form provided by the AIDA Executive.

Evaluation

Evaluation is commonly viewed as being either summative or formative. The distinction serves us well here even if we take some liberties with their definitions:

> Summative evaluation is scientific enterprise designed to test the assumptions underlying the instructional design.
>
> Formative evaluation is an exercise in optimization, designed to determine the best values for unknown parameters of the instructional design.

In most applications, a third aspect of evaluation, quality control, is also required:

> Quality control assesses the conformance of the instructional design and delivery process to its specification.

That these three types of evaluation are interdependent is obvious but often overlooked in practice. A formative evaluation cannot be done until quality-control issues have been resolved. A summative evaluation is meaningless unless the instructional system has been optimized through a formative evaluation.

Quality control. Quality control is based on a specification of system outputs under a particular configuration, without regard to whether that configuration is the optimal or whether the underlying assumptions behind the system are valid. Quality control is achieved when the actual system outputs conform to that specification. Within the AIDA Concept, quality can be assessed by determining the extents to which:

> information provided for content analysis is properly converted to a content representation,
>
> information provided about students, the training environment and the task is properly represented in the SET component,
>
> the AIDA Executive produces specified curricula from SET and Content information, and
>
> the instructional delivery mechanism faithfully executes the procedures produced by the AIDA Executive.

Quality can be assessed in controlled experiments or by natural observation. An appropriate experimental technique for assessing quality relies on the use of a validation suite, a set of standard inputs and outputs known to conform to the system design. Natural experiments for quality evaluation within the AIDA framework are more difficult to design since, in a natural setting, the true values of intermediate stages are unknown. Nonetheless, it may be possible in particular cases to devise indices and standards for quality that are not contaminated by uncertainty in the parameters of the system.

In evaluating quality, it is also important to keep in mind the difference between general, system-related problems and special problems only manifest under certain circumstances. These two aspects are commonly separated by observing system performance under a variety of circumstances. Since special problems in AIDA will almost always be manifest as human error, thorough quality testing calls for observing system behavior with a number different designers, instructors, and other human elements.

Formative evaluation. Formative evaluation is a technique for optimizing system performance under conditions of uncertainty about the system's parameters. Therefore, to understand how formative evaluation might function in the AIDA concept, we need to examine sources of uncertainty in system characteristics. Among such sources of uncertainty are the following.

Indeterminacy in the derivation of content from task. Data from a content analysis will often be insufficient to uniquely determine the cognitive structures that support competent performance. At these junctures, the content analyst makes an arbitrary decision which may have downstream consequences for instruction.

Nonidentifiable parameters of students and environment. It may be theoretically impossible to determine the values of some student or environment parameters at the time of course design and development. Typically some arbitrary choice is made which may, like arbitrary content decisions, have downstream consequences for instruction.

Simplifying assumptions to remove variance. The distributions of students and training circumstances are often unknown at the time of training development. Even if they are known, it is usually infeasible to optimize a design for the entire distribution of possibilities. Typically, then, the designer simplifies the problem by designing for a typical student and typical circumstances. The performance of the system may degrade so in nontypical circumstances that this simplification is not warranted.

Arbitrary assignment of strategy to objective. The mix of strategies used to address an objective is often the result of intuition or isolated empirical results. Strategies other than the ones chosen may, in fact, provide better instruction.

Arbitrary cost-effectiveness decisions. During design, and particularly during delivery, decisions may be made to sacrifice effectiveness for costs savings (or vice versa) without complete knowledge of the tradeoff function. Such practices will often lead to suboptimization of overall cost-effectiveness.

Indeterminacy is assessing student characteristics during instruction. Much interactive instruction relies on assessment-instruction feedback loops in which instruction is tailored to the results of a particular assessment. Procedures for both testing and scoring these instruments may constitute sources of uncertainty.

Formative evaluations have three components. First, specific sources of uncertainty are identified. Second, a sensitivity analysis is conducted to determine which of the identified sources has an effect on instruction. Third, experiments are conducted to evaluate different approaches to the sources that pass a sensitivity criterion.

Formative evaluation is difficult. An inspection of the partial list above indicates the difficulty of even identifying sources of uncertainty. Typically, instructional designers encounter enough difficulty in arriving at one viable set of assumptions that consideration of alternatives is not possible within the resources allotted to the development. Nonetheless, where difficult and potentially critical decisions are made, the three component process can help in selecting the best choice.

Summative evaluation. Summative evaluation can be used to identify where instruction or instructional designs produced under the AIDA Concept meet or fail to meet expectations. A summative evaluation can be as simple as a criterion-referenced test given to students. However, one can envision more thorough summative evaluation designed to assess all stages of the AIDA Concept:

Pretests of students and empirical observations of the training environment can be used to evaluate the validity of the SET Component.

A number of techniques (Anderson, 1988) from cognitive psychology can be used to assess the validity of the Content component as a model of competence.

On occasion, learning models will be available to test the effectiveness of instructional strategies in particular cases, but, as I have pointed out elsewhere Halff (1989), opportunities for this sort of assessment are limited.

Finally, the effectiveness of instructional delivery can be evaluated by tests of student's behavior.

The evaluation model described here serves to illustrate two general points about evaluation.

Performance deficiencies can result from several causes. Implementation can fail to meet design specifications. The design can be suboptimal. The assumptions underlying the design can be in error. Because different remedies apply to these different problems, an effort should be made to separately evaluate their contributions to system performance.

Performance deficiencies can arise at any stage in the development process. For this reason, evaluation efforts should provide information, not only on the nature of problems but also on their specific locus within the AIDA Concept.

No matter how extensive the evaluation effort, these points cannot be ignored. The value of any evaluation instrument lies in its potential to deliver precise information on the cause and locus of deficiencies in the design and development process.

Critical Issues for AIDA

The foregoing material presents a rough description of the AIDA Concept and delineates the issues critical to its implementation in any context. Before turning to context-specific considerations, a review of these issues is appropriate.

Content. The Content component is a cognitive model of what is taught. It has both structural, syntactic aspects and content, semantic aspects. Critical is the syntax governing the structure of course content and, in particular, the extent to which that structure is constrained by instructional considerations. Also critical are the methods used to determine and represent the semantic aspects of the Content component.

SET. The SET Component represents information about students, the instructional environment, and the task being taught. The overriding issue in the design of this component is its relationship to the Content Component. The structure of the content may or may not be reflected in information about students and the instructional environment. Conversely, the structure of the content may or may not reflect the structure of the task. Dependent on the resolution of these questions is the structural complexity of information in the SET Component.

Instructional Strategies. An instructional strategy is a procedure for delivering instruction. One approach to its representation is generative in that each strategy is a sequence of instructional primitives, configured and constrained by grammatical rules. Another approach views strategies as black boxes, which we call transaction frames. Although unstructured, transaction frames are schematic and can be configured to meet any of a broad class of instructional situations. These same issues and approaches apply not only to individual strategies but also to entire curricula.

The AIDA Executive. The AIDA Executive defines the mapping from SET and Content components to instructional strategies. The Executive operates by parsing content into objectives and then matching these objectives to strategies. Identification of objectives is determined by the structure of the content. Information about objective type, students, and environment is influential in determining the strategy.

Instructional Delivery. Instruction is delivered by executing the strategies produced by the AIDA Executive. The level of interaction in these procedures is a major determinant of the information-processing requirements of the delivery system. In addition, since the Instructional Delivery component acts as an interpreter of strategy specifications, the power of the interpreter must be matched to the level of abstraction in the strategy specifications.

Evaluation. Evaluation, within the AIDA Concept, is used to determine where problems exist and how to make improvements. Essential in any evaluation is to determine the cause of problems, be they in quality of implementation, optimality of the design, or validity of underlying design assumptions. Also important is the identification of the locus of any problems, that is, of the particular component of the AIDA Concept that gives rise to the problem.

Resolving all of these issues in general is obviously an impossible problem. In particular contexts, however, many of the issues become irrelevant and others become tractable. To understand how these issues are manifest in particular situations, we turn now to a discussion of the context of instructional design and development. Because a systematic view of instructional development contexts is beyond me, the treatment below is limited to the analysis of two examples of particular development contexts. One of these contexts, the Air Force Technical Training Center, is real although my presentation of that context may depart from reality at several points. The second context, CAI 'R Us, is fictional but representative of a number of instructional development organizations.

The choice of these two contexts was deliberate and intended to focus the discussion on contextual aspects of automation. The next two sections describe the contexts in terms of the AIDA concept and ask how that concept can be profitably automated.

The Air Force Technical Training Center

The Development Environment

The Air Force Technical Training Center is responsible for the design, development and management of Air Force technical training. Training requirements are promulgated by authorities outside of the Center. When training development requests arrive at the Center, they are assigned to one of several departments known as Training Development Branches (TDB). These TDBs are organized around particular technical specialties. Some branches of the Center are involved in special projects such as exportable CAI that cut across technical specialties. Although not explicitly addressed here, these units have much in common with CAI 'R Us, which is discussed in the next section.

When a development request arrives at a TDB, it is placed in the hands of a Training Specialist, who then assumes responsibility for developing the course. These Training Specialists are subject-matter experts in the training to be developed but have little if any expertise in training design and development.

Content. Two mechanisms define the content of courses developed at the center. Requests for training development are accompanied by a document known as Specialty Training Standards (STS). This document, for the most part, defines the structure of the subject matter. The content (as distinct from structure) resides in the mind of the Training Specialist responsible for course development.

SET. The STS defines, in addition to course requirements, the entering capabilities of students. The Training Specialist's personal knowledge is, no doubt, used to supplement this written documentation. Constraints on the training environment (e.g., course length and method of instruction) may be provided with the training request. Otherwise, this information is developed as part of the design process. Task information resides almost completely in the Training Specialist's mind or in technical documentation available to him (Spector, 1990).

Instructional Strategies. Instructional strategies are not explicitly developed in the course of training design and development. Rather, they are implicitly embodied in lesson plans and other course materials. Other aspects of instructional strategy are represented in the instructors' minds.

The AIDA Executive. The Air Force has its version of an ISD model that is meant, among other things, to guide the selection of instructional strategies to meet particular objectives. However, the model is not appropriate to the particular needs of most Training Specialists at the Center, and their lack of training in instructional design renders them ill equipped to implement formal instructional design methods.

I suspect that tradition is the principle determinant of instructional strategies at the Center. Since each of the TDBs develops training within a technical specialty, they accumulate schematic knowledge of instruction in the specialty. In developing training to meet a particular objective, they match the objective to one or another instructional strategy, and configure the strategy to fit the particular objective. Configuration may be largely a process of consulting old strategies that have been previously configured to similar objectives.

Instructional Delivery. Instruction is typically delivered in a conventional classroom setting to classes of about eight students. Students are provided with some written materials and instructors work from written lesson plans. Since a TDB is responsible for both development and delivery of instruction, some aspects of instructional delivery are not explicit in lesson plans or course material but rather are carried directly from development to classroom by Training Specialists. The small class size indicates that the level of interactivity in delivery is quite high.

Evaluation. The principal mechanism for quality control at the Center is documentation of the ISD process. This documentation conforms to a standard Air Force ISD model and is considered more burdensome than helpful in quality control.

Formative evaluation is difficult under most conventional ISD models since opportunities for sensitivity analyses and interim experiments are limited. Although informal evaluation of instruction may feed back into subsequent design, formative evaluation is not a large concern at the Center.

Some attention is given to summative evaluation by including measurement and standards as part of the required ISD process. Measurements are restricted to tests of student performance (usually on pencil-and-paper instruments). These tests might constitute useful summative information were it not for deficiencies in quality control and formative evaluation. There is no systematic way of tracing problems evident in these tests to their sources in the design and development process.

By way of summary, we can point to four aspects of instructional design and development at the center that have particular significance for automating these processes.

First, the division of labor for course development is of interest. The determination of objectives and their sequence is in the hands of authorities outside of the TDB. All other aspects of design, development, and delivery are the responsibility of the Training Specialist.

Second, the population of courses developed at a single TDB or by a single Training Specialist is narrowly circumscribed along technical lines. All courses written within a TDB have roughly the same objectives and address roughly the same technology. What distinguishes one development project from another are the particular technologies involved.

Third, and consistent with the second aspect, those responsible for instructional design and development are subject-matter experts and usually possess no special expertise in instructional science or practice.

Fourth, there is little in the way of automated support for either development or delivery of instruction. Tables and forms used in the ISD process offer the opportunity for some formal representation content and instructional structure. The content itself and most of the instructional procedures are not formally represented. Development and delivery procedures are implemented by the human Training Specialists.

Automated Support for the Technical Training Center

With the understanding developed above of the context governing instructional design and development at the Center, we can envision the kind of automated support appropriate for this context. The following speculations describe first how AIDA might appear to a Training Specialist at the Center and then how this vision relates to the more general AIDA Concept.

Kim and the AIDA: A course development fantasy. Presented here is a brief sketch of course development using AIDA in the TDB devoted to Advanced Carbohydrate Preparation Equipment. The Training Specialist, Kim Cook, is about to prepare a new maintenance training curriculum for the Air Force's latest toaster, the To-14 (including the attack model of the toaster, the To-14A).

The AIDA workstation that supports Kim's development effort has a large screen, appropriate manipulanda and output devices, access to scanners and to the toaster's CALS-compliant technical documentation. The workstation also offers a complete complement of document preparation tools: Optical Character Reading (OCR) software, text and graphic editors, and other useful applications. AIDA itself allows the Kim to operate with any of three views of the evolving course:

A Technical Description view allows her to record and edit information about the toaster itself.

A Training Requirements view allows her to record and track instructional objectives.

A Curriculum view allows her to work with the evolving curriculum.

Kim opens a window exhibiting the top level Curriculum view of the new course. At the moment, it contains nothing but the AF Standard Toaster Maintenance Training Curriculum Template. She also opens a window on the Training Requirements Window loaded with objectives defined in the STS. Finally, she opens a new Technical Description window for the To_14 and a window on an existing Technical Description of the To-12, the direct predecessor to the To-14.

Kim's first task is to create, in the new Technical Description window, a Dynamic Block Diagram (DBD) of the To-14. A DBD is nothing more than a qualitative simulation of the device, and it is entered using an editor similar to that described in Towne and Munro (1988).

Most of the DBD can be copied from the existing To-12 Technical Description. Kim updates the old To-12 DBD with minor technical changes. She also adds the To-14's new Digital Anti-Burn Sensor (DABS) and the explosive-propelled toast-eject unit on the To-14A.

A convention adopted early on in the design of AIDA itself calls for the use of terms related to the technical domains taught at the center or to concrete instructional operations. Training Specialists confused by the term "qualitative model" instantly grasped the meaning of "Dynamic Block Diagram."

As soon as Kim completes the DBD, the Curriculum window changes in almost imperceptible ways indicating that AIDA has "roughed in" certain lessons. She ignores these changes and begins to fill in technical data on the components and connections in the DBD. Some of these data are taken from the To-14's technical documentation, others from the To-12's Technical Description. As these data are added more changes are evident in the Curriculum window.

Kim next turns to a section of the Technical Description known as Standard Maintenance Procedures. She examines a library of toaster-maintenance procedures, chooses those appropriate to the To-14 and, working from the technical documentation, configures each to fit the To-14. She also adds a procedure for testing the DABS, adapting a procedure found in a larger library of maintenance procedures. The new procedure calls for the use of special purpose DABS test equipment, which is automatically added to a list of test equipment to be covered in the course.

With the technical description essentially complete, Kim turns her attention to the curriculum. At this point, the Curriculum window contains a tentative curriculum. Some of the modules, such as the section on standard maintenance procedures are described in some detail. Others, such as troubleshooting training are nothing more than place markers at this point.

Associated with each module in the evolving course are certain training variables, including the time allotted to the module and the media to be used. These aspects of the course were established years ago as part of the To-14 acquisition process. Kim checks these previously established values and revises one to reflect the fact that the modules on standard maintenance procedures will be provided via interactive videodisc since production of the planned maintenance simulator has been held up pending OSHA approval.

Kim then begins a careful inspection of the material itself. A general orientation to the course is to be given in a classroom environment. Kim scrutinizes the lesson plans for that section of the course, fills in the many blanks using material from the technical documentation, and edits the material for coherence and completeness. She adds a discussion of some of the To-14's idiosyncrasies that AIDA failed to include in the course.

The second section of the course provides interactive videodisc practice in standard preventive maintenance and repair procedures. AIDA has already created the computational structure for this training and formulated a proposed videodisc design. Kim edits the material, adding warnings appropriate to some procedures. She uses electronic mail to send the videodisc design to the Center's video production unit along with a request for an initial design meeting.

The last section of the course is a computer-based troubleshooting laboratory. Kim consults the Training Requirements related to troubleshooting and selects a list of target faults for the laboratory. AIDA uses the DBD to create and sequence exercises for each of the faults. It also proposes additional exercises on faults easily confused with those in Kim's target set. She changes the instructional medium for the lab to interactive videodisc and obtains a revised disc design from AIDA. AIDA warns her that the module is too long for the scheduled time, but she ignores this warning since she knows that the students will find enough time to complete the lab.

What is important to understand about this fantasy is that AIDA is designed to talk to the Training Specialist in her terms, that is, in terms of technology and teaching, not in instructional design terms. Kim is never asked to think about facts, rules, concepts, and procedures, or about expository vs. inquisitory strategies, or about learner control. Rather, she is asked what the To-14 is like, what students must learn about it, and what the training environment is like. She is also given control over the end product and the responsibility for aspects of the course beyond the scope of automation.

We are now in a position to ask how each of the components of the AIDA concept can be designed to implement this philosophy.

Content. What makes the foregoing description a feasible picture of automated instructional development at the Center, is the organization of the Center's activities around technical specialties and subject-matter expertise. A typical TDB may, for example, be responsible for the development of maintenance training for each of a class of electronic devices (e.g., radar, avionics). Because maintenance skills are the targets of instruction in each case and because the devices being maintained are all of a kind, the cognitive structures that support skilled performance can be represented in the same fashion in all courses. The initial development of a representational system may be difficult and plagued with the problems mentioned in Section 1.1, but the uniformity of maintenance skills across courses offers the promise that a few common representational systems can serve to represent the content of training in a large number of courses. The content of a typical maintenance course, for example, might employ the following devices:

qualitative models of the type devised by Forbus (1984) to represent equipment functionality.

procedural representations of the type described by Kieras (1987) to represent preventive maintenance and repair procedures. (By repair we mean the procedures needed to correct a known problem, not the fault-isolation procedures used to detect the problem.)

troubleshooting procedures of the type discussed by Hunt and Rouse (1984).

text models of the type discussed by van Dijk (1980) for the representation of technical documentation.

materials relating procedural and conceptual representation to the equipment itself, test equipment, and ancillary materials (simulators, photographs, etc.), perhaps based on some of Baggett's (1985) suggestions.

Some of these representational devices apply to wider range of contexts than equipment maintenance. However, the key to successful representation here is the development of structures that are specialized to maintenance as it is taught in the Air Force. A system for automating the representation of technical documentation should be based not on a general Van Dijk (1980) model but rather on a Van Dijk-like model specialized to represent Air Force technical documentation. A calibration procedure should not be represented within a framework that covers all goal oriented procedures but rather with a schema specific to calibration.

SET. Individuals being trained in similar technological skills are normally drawn from populations with similar backgrounds. Thus, to the extent that courses under the purview of a single TDB address similar technological skills, student and environment variables in one course will usually be the ones that are important in others developed in the same TDB. Furthermore, the treatment of these variables in designing instruction will also be the same across courses.

This is not to say that there is no variation from student to student or training environment to training environment, but simply that the important sources of variation will remain the same from course to course and can be dealt with in the same fashion in different courses. Students deficient in digital logic will be given the same sort of remediation in a course devoted to one sort of radar system as they will in a course on a different sort of radar system. Likewise, the availability of a maintenance simulator for one type of equipment will have the same impact on instruction as the availability of a simulator in a course addressing a different piece of equipment of the same class.

As the consequence, the opportunity exists for determining how to treat SET considerations in, say, a maintenance-training context and automating that treatment implicitly in the AIDA Executive. The general problem of remediating deficiencies or assigning media to instructional objectives is difficult and probably beyond effective automation. Fortunately, these general problems are not of concern to the Training Specialist at the Technical Training Center. Of interest to him are particular students deficient in digital logic and availability of particular maintenance simulators. These specific issues, because they are constrained to the context of maintenance training, can be handled automatically and may even benefit from automation.

Instructional Strategies. Earlier I suggested that the fundamental issue in the development of curricula and instructional strategies was that of generation vs. schema selection. When the space of instructional objectives is highly circumscribed (as it is at the Technical Training Center) and when computers are used to develop curricula, then the latter approach is the one to be favored.

A schema for a maintenance-training curriculum, such as the one described above for the To-14, might have the following main components:

an orientation to equipment structure, function, and documentation;

hands-on training in preventive maintenance and repair procedures; and

a troubleshooting laboratory.

Each of these components will have a structure of its own and will, at the lowest level consist of particular transaction frames. The orientation component, for example, might have a structure dictated by elaboration theory (Reigeluth and Stein, 1983). The procedure-training component might be organized around the occasions for invoking each procedure, but would have an otherwise flat structure. The troubleshooting laboratory could consist of problems that are organized and sequenced in such a way as to promote the development of an effective troubleshooting strategy. The further development of this curriculum for a particular maintenance course could be automated by a mechanism that would sequence and configure transaction frames from a computer model of the equipment in the Content component, material in the equipment's technical documentation, and some material supplied by a Training Specialist.

It is easy to underestimate the importance of restricting the scope of automated curriculum development to a particular domain, in this case, maintenance. To appreciate the importance of this restriction, consider the following:

The top level structure of the curriculum is, itself, specific to maintenance.

Assume one chose to use elaboration theory to structure the orientation component. Automating elaboration theory in general is a major undertaking. Much more feasible is the development of an automated procedure, conforming to elaboration theory, for generating an orientation to the structure, function, and documentation of a piece of equipment.

Many, although not all, of the transaction frames, particularly those for troubleshooting practice, can only be profitably automated as maintenance-specific procedures.

An additional benefit of domain-specific curricula and transaction frames is the possibility, suggested above in our fantasy, of automatically configuring the same instructional strategy to different media. Within a domain, each transaction frame can be assigned a set of alternative media and can be automatically configured to any member of that set. A troubleshooting transaction frame, for example, could be configured to run a qualitative simulator like STEAMER (Hollan, Hutchins, and Weitzman, 1984) or a videodisc-based simulator like GMTS (Towne, 1987). It could also be configured to generate lesson plans for a classroom implementation of a troubleshooting exercise. The possibility of these alternate configurations, however, depends on knowing enough about what is being taught to be able to specify the set of applicable media and the configuration procedure.

To summarize, instructional strategies in technical training can be given a formal representation and, moreover, a representational scheme can be devised to adequately cover the instructional strategies used within a particular technical domain. Strategies represented in such a scheme will need further specialization to particular material, and some may, themselves, be specializations of more general strategies.

However, in practice, AIDA will achieve maximum advantage by representing instructional strategies at the most specialized level that covers the population of courses developed by the Training Specialist.

<u>The AIDA Executive</u>. In the previous section I noted that the task of the AIDA executive is to produce a curriculum of strategies from Content and SET specifications. It also describes a general technique for this task based on decomposition of the content into distinct objectives, assignment of strategies to objectives, and configuration of those strategies to fit the content.

This top-down approach can be feasibly implemented by humans, but presents significant and perhaps insurmountable problems to machines. The source of difficulties in automated top-down planning is that of ensuring conformance of the plan to constraints that cross the boundaries of plan components. In instruction, for example, the inclusion of say an illustration in one part of the course may be critical in selecting and configuring a transaction frame in another part of the course.

The simplest solution to the machine planning problem is to avoid it by providing ready-made plans, and that is the solution suggested for AIDA as described here. The templates and procedures that represent curricula and instructional strategies are nothing more than plans for instruction in particular domains. The AIDA executive, instead of having to reason from abstract instructional principles need only invoke a suite of slot-filling procedures to configure the curriculum and its transaction frames. These procedures, like the curriculum templates and content descriptions would be ad hoc and specific to a domain. For example, a slot-filling procedure for a troubleshooting exercise would first select and sequence faults using a procedure to maintain coherence and then, for each fault chosen, run an optimal trouble-shooting model to generate the correct sequence of troubleshooting actions to isolate the fault.

Instructional Delivery. I see two opportunities for AIDA to advance instructional delivery in settings like the Technical Training Center.

First, AIDA can provide direct support of advanced, computer-based instructional techniques. The information processing requirements of computer-based simulators, intelligent tutoring systems, adaptive testing, and other techniques are such that their implementation requires computer support. A training device such as the IMTS (Towne and Munro, 1988) cannot be feasibly configured for any application without the support of its computer-based authoring facilities.

This is not to say that AIDA in this context should restrict itself to computer-based delivery. One of AIDA's strong points here is that of providing a complete solution to technical training problems. Restricting the media that it addresses to computers would make it inapplicable to any situation requiring the higher levels of interactivity that only "live" systems and human instructors can provide in some circumstances.

A second opportunity for AIDA to advance the delivery of instruction is by helping instructional developers deal with turbulence in training environments and student characteristics. For example, it is seldom the case that all training devices for a piece of equipment are available when the first students need to be trained in the operation and maintenance of that equipment.

A single transaction frame, appropriately configured, should be able to generate equivalent instructional procedures for alternative media. This capability, to reconfigure transaction frames for different circumstances, would greatly ease the burden of producing revised materials as new media become available.

Evaluation. the previous section describes three aspects of evaluation: quality control, formative evaluation, and summative evaluation. The quality-control issues discussed there arise more in connection with the design of AIDA as conceived of here than with its use. Validation of the sort discussed in that section should be carried out on the initial implementation of AIDA and on each of extension to a new technical area. It should also be possible to check the conformance of courses to quality standards such as those found in the IQI (Montague, 1983). Partial support for this validation can be provided by AIDA itself.

AIDA, as sketched in the fantasy above, offers numerous opportunities for formative evaluation. Training Specialists can create different qualitative models of the same device and compare the curricula created by the two models. They can play the same what-if games with other aspects of the design such as media assignment and problem selection. Automated generation of curricula provides a far greater measure of control in empirical comparisons of different design approaches.

Summative evaluation is not particularly relevant to the concept of AIDA as presented here. In the Technical Training Center, AIDA should be viewed as a development tool whose validity should be established prior to implementation. Although AIDA can support the continuous evaluation of students both before and after training, it offers no special advantage in some of the more precise summative evaluation techniques suggested above.

In summary, training development contexts like the Technical Training Center, where training requirements, content structure, and curriculum structure vary little from course to course, are prime candidates for effective automation of instructional design. AIDA, in these contexts, should support instructional design and development through conversations with the developer that address the particulars of the subject matter, the training requirements, and instructional materials. The implementation of this philosophy calls for the representation of both content and instruction at the most specialized level that covers the population of courses under development. Instructional design theory may play a key role in the development of these specialized representations, but it should not play an explicit role in their implementation.

The viability of this conception depends on some nontrivial assumptions of how instruction changes across technical domains and specialties. One way of expressing these assumptions is shown in Figure 8. Technical domains and specialties fall in a hierarchy and close relatives in the instructional hierarchy share more in the way of instruction than do distant cousins.

Thus, for example, the changes needed to adapt To-12 maintenance training to To-14 maintenance training are trivial compared to the changes needed to adopt To-12 maintenance training to a course in ADA programming. This assumption is violated to the extent that small changes in content or training requirements lead to large changes in instructional design. One can expect some such violations but I doubt that they are numerous enough to threaten the viability of AIDA as described here.

The level of generality of AIDA within the hierarchy of Figure 8 is an open question. An AIDA that addresses, for example, only the maintenance training for one particular device is too specialized. An AIDA that addresses all of technical training is too general. I suggest above that AIDA be specialized at the branch level, thus making the question something of an organizational issue. One could envision more general versions of AIDA, for example, one capable of designing maintenance training for any electronic device or maintenance training for all devices. AIDA ceases to become useful, however, when its level of generality exceeds that of the technical knowledge of the subject-matter expert designing the course. A way of having one's cake and eating it too is to rely on a configurable AIDA that could be specialized for development of training in particular specialties. (The computer program responsible for creating specialized AIDAs would be known as the Automated AIDA Design Assistant, or AAIDADA).

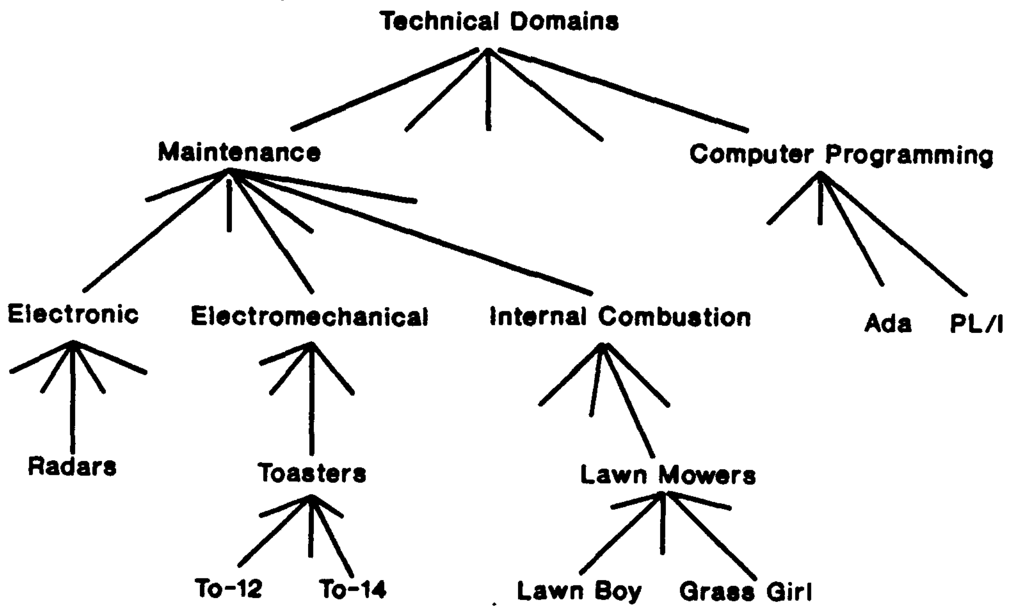

Figure 8. Technical Domains Hierarchy Fragment

CAI 'R Us

The Development Environment

CAI 'R Us is a fictional firm engaged in the development of Computer-Assisted Instruction (CAI). It was founded in 1975 by a group of computer programmers and educators who had formerly worked in a university-based CAI lab. The lab was dissolved due to lack of funding, and CAI 'R Us bought the lab's computer along with rights to the TeachWrite authoring language. TeachWrite, now updated to run on microcomputers, is considered the cornerstone of CAI 'R Us' success. The language's many design flaws -- among them, lack of recursion, block structure, scoping, and structured variables -- are masked by the availability to the author of 2,436 powerful TeachWrite commands.

CAI 'R Us writes courseware mainly for the government or for its contractors. Jobs vary in content but are constrained by suitability of the media. Most of their contracts are awarded through competitive procurement.

Content. Because CAI 'R Us obtains most of its work through competitive procurement and because they concentrate more on development than front-end analysis, they usually start work with a preliminary needs analysis in hand. This analysis, in addition to identifying and structuring instructional objectives, also tailors the scope of the project to the media available to CAI 'R Us.

In spite of the computer-based nature of CAI 'R Us' work, formal representations of the instructional objectives are usually not provided with the front-end analysis. Cursory verbal descriptions of each objective (in behavioral terms) are provided via the RFP, and additional definition is obtained in consultation with subject-matter experts during the course of design and development.

Instructional strategies. In spite of TeachWrite's 2,436 powerful commands, the strategies used in any particular job are small in number. Some of these strategies are common to all of CAI 'R Us' jobs and, indeed, to most conventional CAI. Sequential displays of graphic or text, multiple-choice queries, and so-called interactive, branching simulations are among these common devices. In addition, most jobs require the development of specialized, job specific instructional strategies, usually based on simulation.

The AIDA Executive. The design and development of courseware at CAI 'R Us follows an engineering model. A conceptual design document is developed that describes the curriculum and specifies the strategies to be used for each lesson or unit. A brief description of the content of each unit is also provided with this document. Curriculum design is informal but tied to the instructional objectives provided by the client. A quality control check associates each unit in the design with its associated objective(s) and vice-versa. The selection of strategies is guided implicitly by a few simple rules of thumb such as those found in Merrill (1989). Neither CAI 'R Us nor its clients have ever felt the need to make these rules explicit.

A detail design document is produced after client approval of the conceptual design. This detail design is typically a storyboard description of those parts of the course administered via conventional CAI and a detailed description of any special simulations or strategies. The detail design document then, represents the configuration of strategies selected in the conceptual design phase. This configuration process is, like the conceptual design, informal. Material is written or selected in consultation with subject-matter experts engaged by the client or by CAI 'R Us.

Instructional delivery. CAI 'R Us makes available to its clients the full range of automated instructional delivery devices including interactive videodisc, computer-generated graphics, and a variety of manipulanda (mice, touch-screen, etc.). The level of interactivity is local interactive, to use the terms introduced earlier. Since TeachWrite itself is a general-purpose language with no explicit means of representing instructional content, the translation from detailed design specification to code is done by a humans.

Instruction by humans is not a central part of CAI 'R Us' offerings. The firm does, however, provide some support and training for instructors in automated classrooms.

Evaluation. The typical CAI 'R Us project contains procedures for quality control, for formative evaluation, and for summative evaluation. Clients and CAI 'R Us review both the conceptual and detailed design of the project. Pilot experiments of some or all units allow for formative evaluation. Incorporated into each course are both formal and informal evaluation instruments. Clients are provided with summary statistics of student performance and students' subjective ratings of the course. Although the development context at CAI 'R Us shares some important features with that at the Air Force Technical Training Center, they are different in many ways critical to the issue of automated instructional development and design.

Like the Technical Training Center, CAI 'R Us usually receives something in the way of a front-end analysis from its clients. This analysis provides the firm with instructional objectives that presumably can be met using the media available to CAI 'R Us.

Unlike the Technical Training Center, however, there are no domain restrictions on the projects undertaken at CAI 'R Us. The company may, on one occasion, take a job addressing the training of Social Workers in dealing with drug addicts and, on another occasion, undertake to provide training of budget analysts on electronic spreadsheets.

Also in contrast to the Technical Training Center, subject-matter expertise then is not what CAI 'R Us brings to their work. Rather, they provide expertise in instructional design and development and the development of computer-assisted instruction in particular.

Automation plays a much more critical role at CAI 'R Us than at the Technical Training Center. Since all instruction is delivered by computer, the instructional strategies involved are only those that can be automated. Although these strategies do not themselves have formal representations, they could be formalized and automated with appropriate software.

Automated Instructional Design at CAI 'R Us

We can now try to envision, as we did in the previous section for the Technical Training Center, what AIDA might be like if it were designed for use at CAI 'R Us. As the reader might suspect, the aspects of the CAI 'R Us context just reviewed imply quite a different model of AIDA than the one discussed in connection with the Technical Training Center.

Harry, Mud, and the AIDA: Another course development fantasy. Harry Hylton-Ashcroft, a Senior Design Consultant at CAI 'R Us is working with an RFP from the Department of Energy requesting a self-contained CAI package to train geologists and petroleum engineers in methods for reservoir analysis for use with new secondary-recovery methods. Reservoir analysis is the process of estimating how much oil or gas can be recovered from a prospective or existing oil field. Secondary recovery techniques are those used to recover minerals once the primary flow has been exhausted. Among the most important are water or gas injection and horizontal drilling.

According to the RFP, students need to be able to

collect data needed for reservoir analysis,

select an analysis method based on data available and recovery method,

make use of a DoE-furnished computer program, RESEVAL2, for reservoir analysis, and

interpret the output of the program.

Harry, recognizing that his meagre knowledge of the oil industry extends only to his local gas station, brings in an expert reservoir engineer, Mud Doogan, as a consultant. Harry, Mud, and AIDA work together under a tight deadline to provide a conceptual design for RFP.

Harry's AIDA workstation presents three views of a project:

A SET view permits entry and editing of information about students.

A Content view permits entry and editing of course content.

An Instructional Strategies view permits inspection and editing of the evolving course design.

Harry and Mud begin with the SET view. Harry opens an Enterprise frame and asks Mud, "What, basically, are we trying to teach people to do?"

Mud replies, "Estimate petroleum reserves recoverable using various secondary-recovery methods."

Harry enters Mud's words into the Enterprise frame of the SET view. Harry does not understand what Mud is saying; Mud does not understand what Harry is doing. However, both feel that they have made substantial progress.

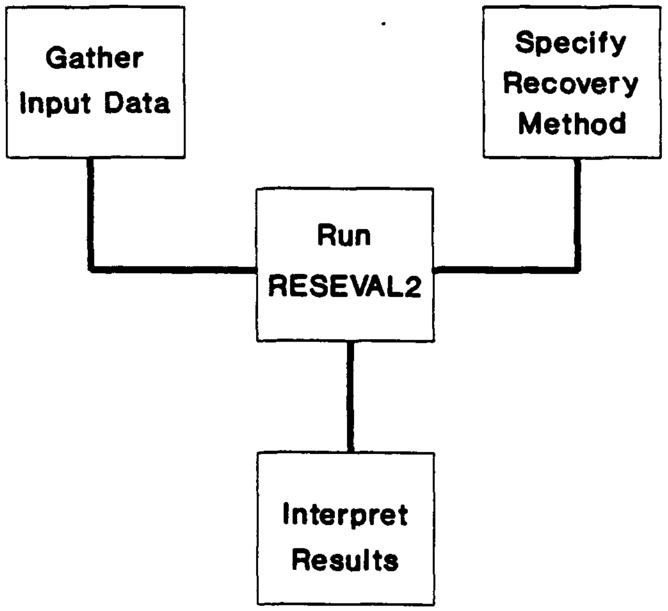

Figure 9. Top-Level Instructional Objectives

Harry then turns to the Content view and begins to elaborate the knowledge needed to succeed in the enterprise. They inform AIDA of a top-level description of the estimation procedure, the result is depicted in Figure 9. Harry then defines the output of the first step depicted in Figure 9, "Gather Data." These data are different sorts (seismic, logs, scouting reports, etc.). AIDA has the capability to represent the structure of each in considerable detail.

Harry is beginning to feel that the job will be a piece of cake, when Mud remarks, "Of course, most of the time, you don't have all that stuff [the data to be gathered]." In the conversation that ensues, Harry learns of some of the complexities in this first step. Some data are always available; some are available at a cost; and some are unavailable at any cost. Depending on the situation, some data are less useful than others. Some data, taken under less than ideal conditions, must be adjusted based on the engineer's professional judgment. Sometimes a first analysis indicates that more data and subsequent analyses are needed. The basis of many of these judgments and decisions is a complex geological model of "what's down there." Harry decides that he cannot respond to the RFP with a complete course on reservoir engineering. He turns his back on AIDA and, in consultation with Mud, formulates a simplified model of the job. Students will be taught how to run the DoE programs and interpret their output under a restricted range of standard conditions and existing data.

Harry and Mud return to AIDA and the elaboration of the first step. They use AIDA's standard forms for describing the data to be gathered in the first step. Mud allows that since gathering the data itself is outside the scope of the course, its preparation for input into the DoE programs is the only aspect of the step that needs to be taught. Harry can use AIDA to represent the data-preparation procedure for some types of data. Others require perceptual skills or complex procedures outside the scope of AIDA's procedure-representation facilities. Harry indicates to AIDA that sample materials can be made available for training and testing.

Moving on to the second step of the process illustrated in Figure 9, Harry learns that the heart of the process is that of specifying the parameters of each secondary-recovery method under consideration. Harry has no trouble in informing AIDA of the structure of the parameters of each possible method and of rules for providing their values. He also finds it easy to specify how the results of this specification should be prepared for input to the DoE programs.

Harry is unable to complete the "Run DoE Program" step in Figure 9, because the DoE programs have not been released yet. However, consultation with DoE reveals that the programs are menu-driven, run on personal computers, and have all of the characteristics of programs for which CAI 'R Us has developed training in the past. He makes a note to include references to earlier work in his proposal.

Moving on to the "Interpret Output" step (see Figure 9), Harry learns from Mud that students will need to learn the output format, the assumptions of the analysis method, and how to assess the accuracy of the analysis. Harry chooses to represent DoE program output as a part-whole structure in AIDA so that students will be taught the components of the output and their location in the output report. AIDA assists Harry in elaborating the analysis methods used in terms of a causal model describing data flow among the program's modules. Finally, Harry uses AIDA's rule-based representational system to construct a procedure for assessing the accuracy of RESEVAL2's estimates. Mud, by this time, is totally overawed and completely confused.

The content analysis is complete enough to warrant some curricular recommendations. AIDA first makes some general curriculum recommendations. In particular, it recommends the five module course depicted in Figure 10.

Both Mud and Harry accept AIDA's top level recommendation. Harry, noting that considerable text and graphic material will be needed to complete the Introduction module, delays further development until after contract award. He asks AIDA to open the second module for further development. AIDA suggests two techniques for teaching students how to format input data. The first is exploratory in that students provide their own data, the second is generative in that sample data are provided to the student. The decision, according to AIDA, should be made on the basis of motivation. It engages Mud and Harry in a series of simple questions about students' jobs and the potential impact of instruction, finally concluding that student motivation is "high." Mud remarks that AIDA apparently doesn't know how petroleum engineers feel about DoE software, so Harry resets student motivation to "low."

AIDA accepts Harry's revised evaluation and his request to generate a sample of the problems that students would be given to practice preparing data for input to RESEVAL2. AIDA reminds Harry that some graphic data (sample electronic log records) have not yet been provided. It then presents a few sample exercises from the module just as they would appear to students. Mud is impressed; he is particularly impressed by one problem in which students are given data from a Sprayberry well that yielded half a million barrels of oil. (The term "Sprayberry" refers to a formation found at about 7,000 feet. Wells drilled in this formation normally produce about 70,000 barrels during their lifetime.)

Harry, realizing that not all combinations of inputs are realistic, assesses his options for obtaining reasonable problems: develop off-line, realistic data sets, construct or purchase a geological model that will generate plausible values, or construct an approximate rule-based generator within AIDA that will generate realistic data for perhaps 95% of the problems. He decides that the last choice is the least costly and makes a note to cost out the development of the rules.

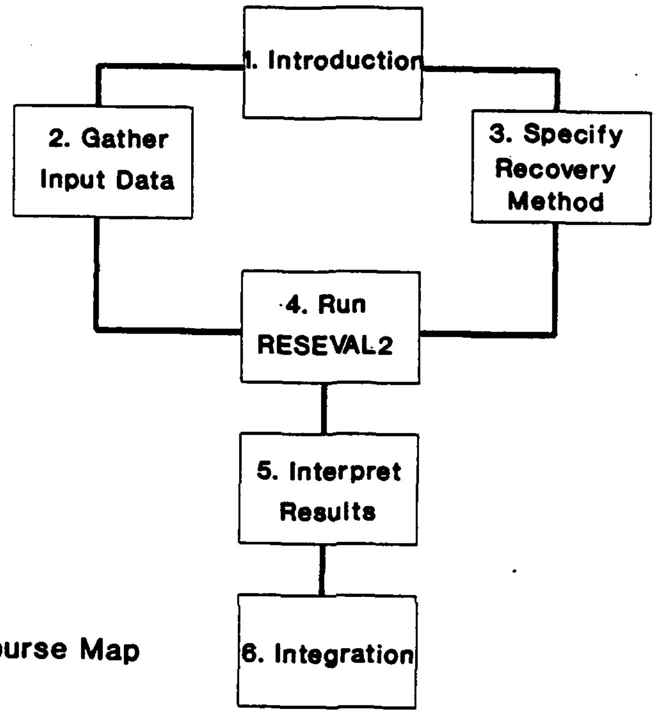

Figure 10. Course Map

Harry and Mud continue their work with AIDA and, within a short time, are able to generate a conceptual design for the project and enough sample materials to support a response to the RFP. Although Module 3 is not in the design (because RESEVAL2 is not complete), the design for Module 4 is quite impressive. AIDA generates an entire interactive lesson on different parts of RESEVAL2's reports. Also part of the design is a hypertext facility that students can use to inspect the methods used by RESEVAL2 to generate each result. Finally, AIDA automatically designs a lesson with integrated practice that teaches students how to assess the accuracy of the figures produced by RESEVAL2. Since CAI 'R Us won the contract and were, with Mud and AIDA's help, able to finish the work on time and under budget, a summary of AIDA's contribution to the project is worth mentioning.

AIDA took CAI 'R Us from an enterprise that depended primarily on hand-crafted instructional strategies to one in which machines are responsible for a large part of curriculum design and development. This transformation was accomplished by explicitly automating all of the strategies previously used only implicitly at CAI 'R Us (plus a few more) and interfacing the strategies thus automated to collection of knowledge representation mechanisms that cover most of the content of interest to CAI 'R Us' clients. Representation of some material is awkward at best, and not all content can be represented in the system, but enough content goes directly from AIDA into courseware to have a major impact on the design and development process at CAI 'R Us.

How AIDA might be designed for effectiveness in the CAI 'R Us development context is taken up in the rest of this section.

Content. The nature of CAI 'R Us' business dictates the criteria that apply to representation of its content. AIDA must provide generality in the sense of being able to represent content from a number of different subject domains. It must provide parsimony in using as few knowledge-representation devices as possible. It must provide knowledge structures that are useful in instructional design. These three constraints cannot all be met by the same system.

As a compromise, AIDA offers one knowledge-representation mechanism for each of three major classes of knowledge:

Production systems are used to represent procedural knowledge.

Schema are used to represent declarative knowledge

Qualitative-process theory (Forbus, 1984) is used to represent causal knowledge.

AIDA also provides, at the most concrete level, for the inclusion of uninterpreted text, graphics, and computer programs.

Because these mechanisms are general, they are useful in the representation of much of the knowledge in each of CAI 'R Us' projects. They fail when specialized knowledge, such as geology, is too complex to be feasibly analyzed within the scope of the project. The three mechanisms are obviously parsimonious since they are almost completely unencumbered with ad hoc mechanisms.

In the initial version of AIDA, knowledge representation was completely unencumbered with special-purpose mechanisms. Later versions have incorporated certain mechanisms, such as iteration, that are found in almost every application.

Perhaps most important are the availability of instructional strategies that can be directly linked to the knowledge structures so that they are useful in designing instruction.

Content representation in AIDA are further constrained by a decomposability requirement. The content of a project must have a hierarchical representation since the AIDA executive (described below) decomposes the content along hierarchical lines.

SET. Recall that in Section 1.2, we discussed two approaches to representing information about students, environment, and task. One approach is to use a fixed vector of features, such as motivation, to represent these aspects of the instructional situation. A second approach provides more precise information by using structures in the SET component that reflect the structure of the content. This latter approach would, for example, support an intelligent tutoring system that concentrates instruction on a student's individual weaknesses.

AIDA at CAI 'R Us uses the former, simpler model. Data on student features are gathered as needed to select instructional strategies, as we saw above when AIDA's recommendation of a generative strategy depended on student motivation. Information about the instructional environment is handled in the same way. For example, certain strategies are not recommended if an instructor is not available for consultation. AIDA's representation of the task is even more simplistic, consisting of nothing more than an uninterpreted statement of the overall goal of training. AIDA relies on this statement to orient students to the training situation.

Instructional Strategies. AIDA's instructional strategies at CAI 'R Us are similar to those described above in Section 2.2 in connection with the Technical Training Center. A collection of instructional schemata are available within AIDA. Each of these schemata, once configured, becomes a procedure that implements a particular instructional strategy. The schemata are specialized to different levels of abstraction. That is, some schemata are transaction frames that embody particular interactions with students. Others handle lessons or collections of transaction frames. Still others handle entire curricula. Configuring a schema is a matter of drawing on appropriate subject matter from the Content module, selecting subschemata, and selecting media.

The AIDA Executive. In contrast to the Technical Training Center, AIDA at CAI 'R Us is a top-down course designer. AIDA's knowledge of how to teach is embedded in a system for matching schemata to elements of the content hierarchy. The main criterion for matching is the correspondence between the knowledge structure to be taught and the candidate strategy. That is, each strategy knows the kind of knowledge that it can teach. More general strategies match objectives at higher levels of the content hierarchy, and more specific strategies match low-level elements of the hierarchy. For example, a general curriculum strategy, shown in Figure 9, matched the first-level content decomposition shown in Figure 10. a specific hypertext transaction frame matched the specific content element representing RESEVAL2's models and assumptions.

In most cases, conflicts will arise when several strategies match a content element. The converse situation, an impasse where no strategy matches a particular element, is one about which we do not like to think.

AIDA uses three mechanisms to resolve these conflicts. If the element being matched has descendants, AIDA can often eliminate strategies that cannot be matched at lower levels. Decisions among those remaining can often be made using rules of thumb that reference student and environment features. As a last resort, the human developer can be asked to resolve the conflict.

Instructional Delivery. One of the key's to AIDA's success at CAI 'R Us is the restriction of instructional delivery mechanisms and the consequent restriction of instructional strategies. CAI 'R Us does not deal with instructor-delivered instruction or it's computer-based equivalent. Hence, it has no need to consider or represent the dialog structures required to implement this sort of instruction. Nor does CAI 'R Us deal, in any serious way with non-interactive media such as text and videotape. These media are used in CAI 'R Us' instructional system, but their structure is never a design problem for AIDA; AIDA never generates text or a shooting script for a videotape. Rather, the delivery mechanisms available for use at CAI 'R Us are those that achieve the middle levels of interactivity described in an earlier section.

Of interest is the fact that these middle levels of interactivity are fairly new to the instructional enterprise. Conversation and discourse have been with the human race for tens or hundreds of thousands of years. As the consequence they may have a complexity that has heretofore defied all attempts at automation. Human-computer interaction is newer to the human scene and is governed by simpler and less elaborate structures. It is this simplicity that renders feasible the automated design of computer-based instruction.

Evaluation. AIDA at CAI 'R Us offers considerable opportunity for evaluation. Because AIDA itself undertakes much of the design, quality control issues arise mainly in connection with the assistance provided by human instructional developers. Quality can be assessed in experiments using standard course materials, and defects can be traced to specific points in the design and development process. Naturalistic assessments are also possible by sampling materials or decisions provided by instructional designers and assessing the quality of the sample. For purposes of formative evaluation, most "degrees of freedom" in a project occur in the design of the Content component and the choice of instructional strategies. AIDA can support the evaluation of various options in each cases by producing designs that implement each option. The resulting designs can be submitted to a sensitivity analysis, and the ones that differ in significant ways can be implemented and tested empirically.

Three aspects of summative evaluation are pertinent to AIDA as it functions at CAI 'R Us. Of primary interest is the scope of AIDA's capabilities. Some courses can be designed and developed using AIDA's standard capabilities. For others AIDA serves as little more than a shell for ad hoc instructional methods implemented outside of its normal mechanisms. In some cases these failures will be those of the Content module to represent the subject matter. In others failure can be traced to lack of a suitable strategy. In all cases it is important to examine where, in each course or project, AIDA fails to provide an appropriate representational or instructional method.

Second, there will be cases in which AIDA has the potential for designing adequate instruction but in which design proves impossible because the system is simply unusable. Failures of this sort can be traced to low-level developer-AIDA interface problems, to the lack of suitably strong mechanisms for developer-AIDA interaction, to lack of training, or to a host of other issues.

Finally, there will be cases in which design tasks are well within AIDA's capabilities, and no problems arise in implementing the design, but the instruction produced is simply not effective. These problems will be manifest in performance measures taken from students after all other sources of deficiencies have been eliminated.

In summary, the AIDA found at CAI 'R Us is quite a different beast than that found at the Air Force Technical Training Center. CAI 'R Us brings a relatively restricted set of instructional mechanisms (those available through CAI) to bear on an almost unrestricted content population. AIDA therefore offers general mechanisms for representing subject matter that can accommodate a large portion of the subject matter presented to the firm in a way that offers considerable leverage on the design of instruction. Instructional strategy selection within CAI 'R Us' AIDA relies on explicit selection rules and top-down approach to instructional design. These commitments are required to handle the range of subject matters of interest to CAI 'R Us and are feasibly implemented because of the restricted range of instructional mechanisms available to the firm. Because design principles are explicit in CAI 'R Us' AIDA, evaluation can be quite precise.

Also worth noting is the nature of the interaction between developer and AIDA. In contrast to AIDA at the Technical

Training Center, AIDA at CAI 'R Us organizes the design process around an explicit model of instructional development. Developers delineate instructional objectives and have direct control over the entire design process.

Automating the AIDA Concept

This concluding section reviews the issues that must be faced in designing AIDA, summarizes its design goals, reviews the design concepts needed to meet these goals, and identifies unresolved issues.

As I hope the reader has noticed, however, the design goals of AIDA, the means of reaching those goals, and the issues associated with the design, depend critically on the implementation context. Indeed, if the reader has grasped only this point, this paper will have served its purpose. This paper therefore needs at least two conclusions, one for each of the contexts discussed above.

AIDA at the Air Force Technical Training Center

In both this section and the next, we will address four questions. What characteristics of the development context are critical to AIDA's design goals? What should AIDA be like in the context? What kind of design will satisfy the design goals? What research is needed to arrive at a design?

Characteristics of the context. Recall that the Technical Training Center is organized along technical lines. Each TDB is responsible for training in a narrowly defined technical area. Courses are designated by outside authority to meet specified primary objectives. Subject matter experts, unskilled in instructional systems design, are responsible for course design and development. Courses are delivered in lecture-discussions to small classes, but other media can be and are employed.

Design goals. AIDA will function best in this context if it is aligned to the mission and experience of its users. It should be conversant in the technology being taught and be prepared to create instruction from technical descriptions of the subject matter. AIDA's recommendations should conform to the principles of effective instructional design but its users should not be required to be conversant in those principles. Rather, AIDA should provide, to its users, the specialized representations of content and instruction that promote efficient automatic generation of instruction and control over the final instructional product.

Design. These design goals can be met by developing, for each subject-matter domain:

specialized curriculum templates appropriate to the domain or specialty,

both specialized and general instructional strategies addressing particular skills known to be critical to course objectives (but which can be configured for different media), and

content representations designed to provide material needed to irplement the curricula and instructional strategies.

The AIDA Executive should operate in a bottom up fashion. That is, it should start with a curriculum template for the course, select instructional strategies and then configure those strategies from the Content component.

Research issues. The most critical research issue for AIDA, as conceived of here, is the nature of the hierarchy illustrated in Figure 6. Needed is an understanding of how instruction becomes progressively constrained as content becomes progressively more specialized. Although this issue can be partially illuminated by analytical means, instructional analyses of actual or proposed courses will be needed for a complete resolution.

Once Figure 8 has been mapped out for a course population of interest, and once it is known how instruction is successively constrained as one moves down the hierarchy, one can resolve certain critical design issues for AIDA:

circumscribing the subject-matter domain(s) to which particular versions of AIDA apply,

determining the instructional strategies and media needed for effective instruction in these domain(s),

formulating the knowledge structures that support the configuration of instructional strategies, and

developing the conceptual and computational mechanism for the generation of multiple specialized versions of AIDA.

AIDA at CAI 'R Us

Characteristics of the context. Unlike the Technical Training Center, CAI 'R Us must develop courseware for a wide range of subject matters. However, the media and instructional methods available to them are limited to those that can be implemented with conventional computer-based media. Development personnel are available that have considerable skill in all aspects of instructional design, but subject matter experts are supplied by clients or engaged as consultants.

Design goals. AIDA at CAI 'R Us must meet two main objectives. First, it must provide a means of constructing curricula for a wide range of subject matters. Second, it must implement these curricula using the mechanisms available through computer-based instruction. Because of its general scope and because CAI 'R Us relies on experienced developers, AIDA should speak the language of instructional design, make explicit its design decisions and their basis, and give developers control over the design process.

Design. To accommodate these design goals, AIDA should be based on:

> a set of instructional strategies that define the course at various levels ranging from a top-level decomposition of the curriculum down to individual transaction frames,
>
> transaction frames that are tailored to the methods and media available with conventional computer-assisted instruction, and
>
> general knowledge-representation mechanisms for representing content.

The AIDA executive should function in a top-down manner. It should use a production system or a similar type of pattern-matching mechanism to select and assemble instructional strategies, starting with a general plan for the curriculum and proceeding to successively lower levels of detail.

Research issues. Three major issues must be at least partially resolved for AIDA, as conceived of at CAI 'R Us, can be developed in a systematic fashion.

The first of these issues is the domain of application of AIDA. Defining this domain as whatever might come to CAI 'R Us for development or even whatever is amenable to instruction via conventional CAI may be sufficient for the purposes of this exposition, but it is not sufficient to support the actual design of AIDA. If the boundaries and structure of AIDA's domain are not defined, then design decisions will be made arbitrarily to support whatever applications are used for development and validation. Specialization, of the type made explicit in the Technical Training Center's AIDA, will occur implicitly in the development of CAI 'R Us' AIDA. Design decisions made prematurely on the basis of a limited sample of applications will face reconsideration or hacking under stress from other applications.

The second issue is that of knowledge representation. For purposes of exposition, we proposed a small number of general knowledge-representation mechanisms (see Section 3.2). Although such mechanisms might, in principle, be adequate for a broad range of applications, in practice, the complexity of many domains will render them infeasible for representation with a few general mechanisms. However, specialization of knowledge-representation mechanisms entails specialization also of instructional strategies. Carried too far, one might find oneself with all of the disadvantages of a specialized AIDA and none of the advantages.

The third issue is that of the instructional design process itself. As was mentioned in Section 2.2, top-down planners face the problem of incorporating constraints across widely separated elements of the plan. The solution to this problem that we suggested above is by no means new and involves the hierarchical refinement of the instructional plan using strategies pitched at various levels of the hierarchy. Needed to implement this approach is a collection of strategies that is broad enough to cover all applications without reaching impasses in the design process and yet specific enough to usefully constrain the course design.

To call these issues "research issues" is something of a misnomer for I see no way of resolving them through a systematic research program. Rather, their resolution will lie in a combination of arbitrary decisions and cut-and-try R&D.

Conclusion

Offered here are two very distinct approaches to the functions and design of AIDA. Both approaches conform to my bias that AIDA should be more of a tool in the hands of a course developer than a development engine. Because of this bias, I recommend that developers with different backgrounds working in different contexts be given different types of tools, even if they have similar short-term goals. A subject matter expert developing instruction in a context defined by his technical specialty needs a tool that talks to him in technical terms and isolates him from unnecessary instructional considerations. An instructional designer working in a context defined more by instructional technology needs precise and explicit control over most instructional-design decisions and a system that does not restrict him to particular content domains.

Other instructional development contexts, not considered here, will impose their own constraints on AIDA's design goals.

For example, much training is so resource constrained that the typical instructional activity must address multiple instructional objectives and the typical instructional objective is addressed by multiple instructional activities. Meeting this requirement is a challenge (to say the least) for any designer of the AIDA Executive.

As another example, instructional objectives in many contexts defy formalization but are nonetheless critical to instruction. The success of a course may, for example, depend on the extent to which it conveys general corporate goals to the students. These goals are usually so general that they admit of no useful computer representation. However, they constitute a powerful influence in helping students deal with particular problems. In these and other contexts with ill-defined instructional problems, AIDA may be completely useless.

Those pursuing the development of AIDA must be aware that it will have different functional specifications in different contexts. This awareness may permit the judicious choice of one or more contexts for further research and development, or, at the least avoid the disaster of developing AIDA for one context and evaluating it in another.

IV. CONCLUSION (Spector)

In Section III Halff provided a view of how an Advanced Instructional Design Advisor could support the development of two broad instructional paradigms. Halff argued that cognitive structures have a significant role to play in instructional design. Learning mechanisms and instructional design are interdependent, a fact overlooked by many systems.

As part of his vision, Halff described two approaches to the automation of instrucitonal design: advisory and generative. The advisory approach attempts to automate the formulation of instructional designs, making implementation primarily the responsibility of a human developer. The generative approach attempts to use the computer to generate instruction from known instructional paradigms. He then presented two examples of generative approaches and argued that generative approaches showed some promise. He argued that successful automation of instructional design would require a prudent division of labor between the human and the computer. Human developers would still be required to make decisions concerning the interpretation of instructional design principles.

In Section IV Halff reviewed the functional components of an Advanced Instructional Design Advisor and indicated some of the problems that might be encountered in each area. The representation of instructional strategies is a crucial issue for an Advanced Instructional Design Advisor. Strategies could have a procedural representation or a schematic representation, depending on the type of Advanced Instructional Design Advisor desired. Procedural representations could relieve developers of the burden of knowing a lot about instructional design; that conception was elaborated in Halff's TTC example. Schematic representations are highly configurable but require the developer to know a lot about instructional design; this conception was elaborated in the CAI 'R US example. Each type of Advanced Instructional Design Advisor is possible, although the concerns, problems, and applicability of each type are quite different.

Halff also stressed the importance of building evaluation into each of the components of any type of Advanced Instructional Design Advisor. This same concern has been elaborated by Tennyson in Volume 2 of this series.

The value of Halff's work is that it provides clear choices and approaches for the automation of instructional design. Halff also provides research issues and potential problem areas for particular choices and approaches. As Halff has remarked on several occasions, it will be important to clearly specify the potential users and to involve them in the design, development, and evaluation of an Advanced Instructional Design Advisor.

REFERENCES

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds.), Foundations of intelligent tutoring systems (pp. 21-53). Hillsdale, NJ: Lawrence Erlbaum Associates.

Baggett, P., & Ehrenfeucht, A. (1985). A multimedia knowledge representation for an "intelligent" computerized tutor (Tech. Rept. No. 142). Boulder, CO: University of Colorado, Institute of Cognitive Science.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18(1), 32-42.

Brown, J. S. & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. Cognitive Science, 4, 379-426.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems (pp. 157-183). New York: Academic Press.

Carr, B. & Goldstein, I. P. (1977). Overlays: A theory of modeling for computer-assisted instruction (AI Memo No. 406). Cambridge, Massechusetts Institute of Technology.

Collins, A. & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. In R. Glaser (Ed.), Advances in instructional psychology (Vol 2, pp. 65-119). Hillsdale NJ: Lawrence Erlbaum Associates.

Crawford, A. M., & Hollan, J. D. (1983). Development of a computer-based tactical training system (Spec. Rep. NPRDC-SR-83-13). San Diego, CA: Navy Personnel Research and Development Center. Dawkins, R. (1976). The selfish gene. Oxford: Oxford University Press.

Dreyfus, H. L. (1972). What computers can't do. New York: Harper & Row.

Forbus, K. (1984). Qualitative Process Theory. Artificial Intelligence, 24, 85-168. Gagne, R. M., & Briggs, L. J. (1979). Principles of instructional design. New York: Holt, Rinehart & Winston.

Gagne, R. M., & Briggs, L. J. (1979). Principles of instructional design. New York: Holt, Rinehart & Winston.

Gold, E. M. (1967). Language identification in the limit. Information and Control, 10, 447-474.

Greeno, J. G. (1989). A perspective on thinking. American Psychologist.

Halff, H. M. (1989). Prospects for automating instructional design. Arlington, VA: Halff Resources, Inc. Hunt, R. M., & Rouse, W. B. (1984). A fuzzy rule-based model of human problem solving. IEEE Transactions on Systems, Man, and Cybernetics, SMC-14, 112_120.

Hollan, J. D., Hutchins, E. L., & Weitzman, L. (1984). Steamer: An interactive inspectable simulation-based training system. AI Magazine, 5, 15-27.

Hunt, R. M., & Rouse, W. B. (1984). A fuzzy rule-based model of human problem solving. IEEE Transactions on Systems, Man, and Cybernetics, SMC-14, 112_120.

Kieras, D. E. (1987). The role of cognitive simulation models in the development of advanced training and testing systems (Tech. Rept. TR-87/ONR-23). Ann Arbor, MI: University of Michigan, Technical Communication Program.

Lave, J. (1988). Cognition in practice. New York: Cambridge University Press.

Mehan, H. (1979). Learning lessons: Social organization in the classroom. Cambridge, MA: Harvard University Press.

Merrill, D. M. (1989). Project AIDA: A concept paper. Logan UT: Utah State University. Reigeluth, C. M., & Stein, F. S. (1983). The elaboration theory of instruction. In C. M. Reigeluth (Ed.), Instructional design theories and models: An overview of their current status (pp. 335-381). Hillsdale, NJ: Lawrence Erlbaum Associates.

Montague, W. E. (1983). Instructional Quality Inventory. Performance and Instruction, 22 (5), 11-14.

Rouse, W. B., & Hunt, R. M. (1984). Human problem solving in fault diagnosis tasks. In W. B. Rouse (Ed.), Advances in man-machine systems research (Vol. 1) (pp. 195-222). Greenwich CT: JAI Press.

Rouse, W. B., Rouse, S. H., & Pellegrino, S. J. (1980). IEEE Transactions on Systems, Man, and Cybernetics, SMC-10, 366-376.

Spector, J. M. (1990). Designing and Developing an Advanced Instructional Design Advisor (AFHRL-TP-90-52). Brooks AFB, TX: Training Systems Division, Air Force Human Resources Laboratory.

Towne, D. M. (1987). The generalized maintenance trainer: Evolution and revolution. In W. B. Rouse (Ed.), Advances in man-machine systems research (Vol. 3, pp. 1-63). Greenwich CT: JAI Press.

Towne, D. M., Johnson, M. C., & Corwin, W. H. (1983). A performance-based technique for assessing equipment maintainability (Tech. Rep. 102). Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California.

Towne, D. M. & Munro, A. (1988). The intelligent maintenance training system. In Psotka, J., Massey, D. L., & Mutter, S. A. (Eds.), Intelligent tutoring systems: Lessons learned. Hillsdale, NJ: Lawrence Erlbaum Associates.

Towne, D. M., Munro, A., Pizzini, Q. A., Surmon, D. S., & Wogulis, J. (1988). ONR final report: Intelligent maintenance training technology (Tech. Rep. 110). Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California.

van Dijk, T. A. (1980). Macrostructures. Hillsdale, NJ: Lawrence Erlbaum Associates.

VanLehn, K. (1987). Learning one subprocedure per lesson. AI Journal, 32, 1-40.